(NASA-CR-144094) MIRADS-2 IMPLEMENTATION MANUAL (Computer Sciences Corp.) 155 p HC $6.75 CSCL 05B

N76-13953

Unclas
G3/82 06066

MA-006-002-2H

GEORGE C. MARSHALL SPACE FLIGHT CENTER

MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM

MIRADS-2 IMPLEMENTATION MANUAL

JUNE 1975

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

SECTION 1 - INTRODUCTION

1.1 GENERAL

The Marshall Information Retrieval and Display System (MIRADS) is a Data Base management system designed to provide the user with a set of generalized file capabilities. These capabilities allow the user to create a Data Base and define its logical structure to the MIRADS System. Additionally, the system provides a wide variety of ways to process the contents of the Data Base and includes capabilities to search, sort, compute, update, and display the data. The process of creating, defining, and loading a Data Base is generally called the "loading process." The purpose of this manual is to define the steps in the loading process which includes (1) structuring, (2) creating, (3) defining, (4) and implementing the Data Base for use by MIRADS.

The organization of this manual is generally in a front-to-back manner, taking the user from the first step in loading a Data Base to the last. However, the execution of several computer programs is required to successfully complete all steps of the loading process. These programs and all other programs necessary for implementation reside in the MIRADS System Library. This library must be established as a cataloged mass storage file as the first step in MIRADS implementation.

The procedure for establishing the MIRADS Library is given in Section 8 of this manual.

The system is currently operational for the UNIVAC 1108 computer system utilizing the Executive Operating System. All procedures within this manual relate to the use of MIRADS on the U-1108 computer.

SECTION 2 - HOW TO STRUCTURE A MIRADS DATA BASE

2.1 DATA STRUCTURE TERMINOLOGY

Paragraph 2.1 defines the terms used in this manual to describe the organization and structure of the MIRADS Data Base. There are four basic terms that are used to represent various levels for the collection or assimilation of data. In a scale ranging from lowest to highest, they are Data Element, Record, File, and File Set.

1. Data Element

   The basic unit of named data in a Data Base is a data element or data field. In addition to a name, a data element has other characteristics pertinent to its use in a Data Base. For example, a data element may be named CITY and have values of New York and Los Angeles.

2. Record

   A record is a named collection of data elements of related information which can be identified as a particular record type, and is viewed as a contiguous collection of data by MIRADS. There may be an arbitrary number of occurrences in the Data Base for each record type that has been defined. For example, there would be one occurrence of the CITIES record for each CITY in the Data Base.

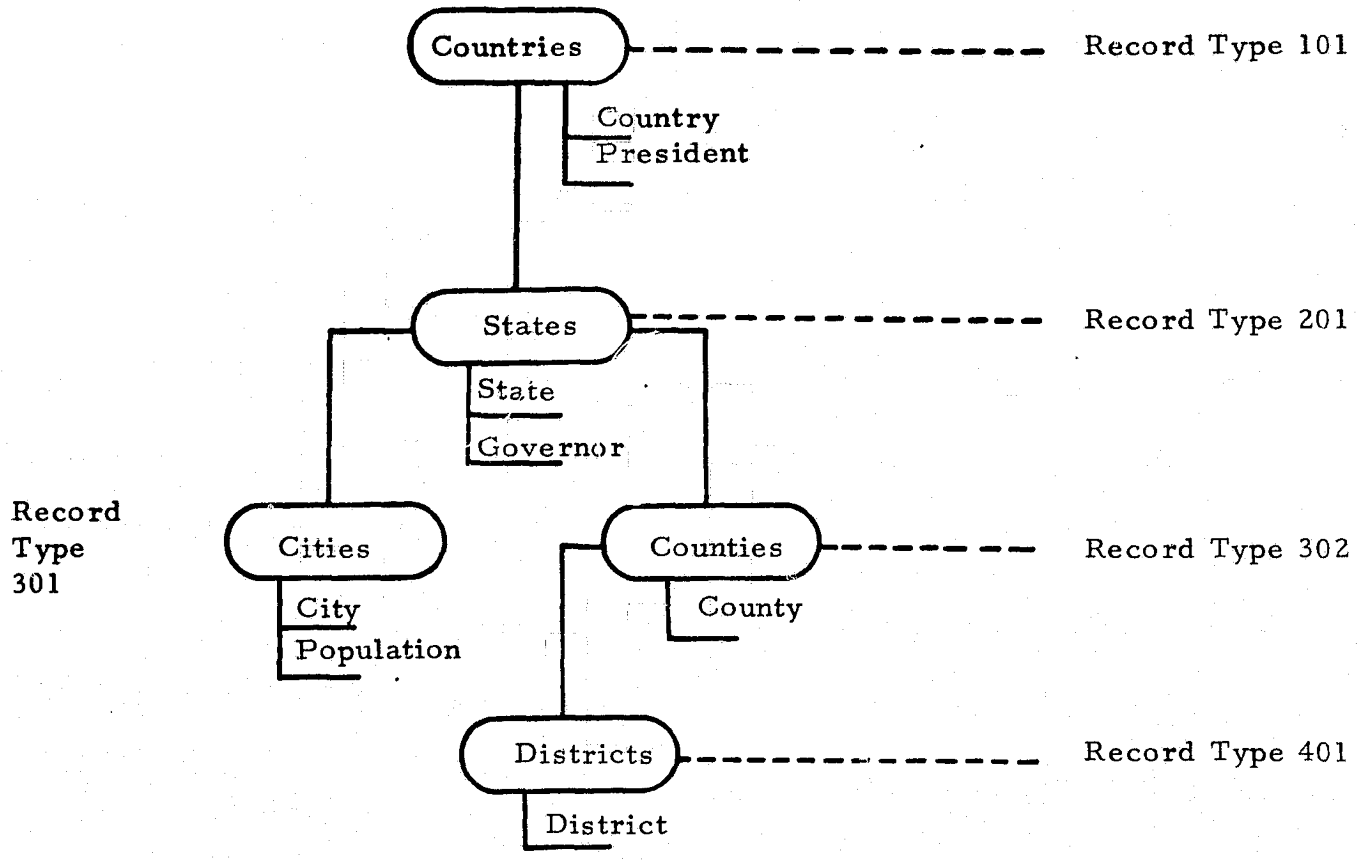
Records may vary in frequency, format, and their relationship to each other. The relationships of records are either subordinate, peer, or superior with respect to other records. In Figure 2-1, the CITIES and COUNTIES records are subordinate to the STATES record, but they are peer of each other. The STATES record is superior to the CITIES and COUNTIES records, but subordinate to the COUNTRIES record.

3. <u>File</u>

A file is a named collection of all record occurrences which have the same logical organization. Figure 2-1 is an example of the NATIONS Master file. It consists of data entries for all countries, states, cities, counties, and districts which compose this application Data Base.

4. <u>File Set</u>

A file set is a collection of several related files, and is used in MIRADS to describe the files of data and control information necessary to implement a user's particular application. This data includes a set of five related files named Master file (MAS), Dictionary file (DIC), Data Relational List file (DRL), Index file (IND), and Save-Query-Set file (SAV). Collectively, these five files are referred to as the MIRADS User's File Set.



NATIONS FILE

Record Type	Descriptor	Relationship
101	COUNTRIES	Superior to Record Types 201, 301, 302, and 401
201	STATES	Subordinate to Record Type 101; Superior to Record Types 301, 302, and 401
301	CITIES	Subordinate to Record Types 101 and 201; Peer to Record Type 302
302	COUNTIES	Subordinate to Record Types 101 and 201; Peer to Record Type 301; Superior to Record Type 401
401	DISTRICT	Subordinate to Record Types 101, 201, and 302

Figure 2-1. Logical File Structure

The portion of the File Set that constitutes the user's Data Base is the Master file (MAS). This file is referred to as the user's Data Base throughout this manual, and it is this file that the user must be concerned with in regard to its organization, structure and storage.

2.2 MIRADS FILE STRUCTURE

This paragraph describes the file structure and organization of the physical structure or layout of the data, and its indexing scheme. Each of these three areas is discussed briefly, followed by a series of questions and answers which explain in detail those capabilities supported by MIRADS. Several of the questions and answers refer to the logical file structure shown in Figure 2-1 for a sample Data Base named the NATIONS file. This file organization was purposely chosen to illustrate pertinent points regarding the structure of any MIRADS Data Base, and is used throughout this manual for explanatory purposes. It does not necessarily constitute the best logical organization for this Data Base. This organization is just one of several which might have been chosen depending on the relationship of the data as defined by the user.

2.2.1 Logical File Organization

The logical file organization is the method used to collect and organize data. Related data elements are collected into groups called records, and related groups or records are assembled into logical files. Within a logical file, records may be ordered according to their relationship to each other, i.e., superior, peer, or subordinate record.

The logical file organization of the MIRADS Data Base is defined as a hierarchically structured file (commonly called a tree structure) which contains a maximum of seven levels of file subordination. The hierarchy starts with a single record at the base of the structure called the master record. From the master record or any other record, one or more records may branch out. However, a given record may have only one immediately superior record. Records at the same level in the tree structure are peers of each other. The following questions and answers provide more insight into the logical organization of the MIRADS Data Base.

1. How many levels of file subordination can exist for a given logical file? The following example represents three levels of subordination:

```
COUNTRIES
        STATES
            COUNTIES
                    DISTRICTS
```

   ANSWER: The MIRADS logical file structure will allow a maximum of seven levels of the file subordination. The COUNTRIES record type in the above example is referred to as the Level-1 record, the STATES record type as the Level-2 record, etc. Using this terminology, MIRADS supports Levels 1 through 8, which is seven levels of file subordination.

2. Does a MIRADS Data Base have to be hierarchically structured?

   ANSWER: No. The user proceeds with Data Base creation and definition for the simple Data Base just as is done for the hierarchically structured Data Base. However, the process is much easier because no consideration has to be given to file structuring and organization. The simple Data Base is a special case of the hierarchical organization where there are no levels of file subordination. The user merely has to create the Data Base using the MIRADS IOPKG, define it using the MIRADS Dictionary, and complete the load process to make the data available for on-line or batch use.

3. How many data elements can exist for a given logical file?

   ANSWER: The maximum number of data elements that can be included in a logical file is 10,000.

4. How many data elements can exist for a given record type? In the example in Figure 2-1, the CITIES record contains two data elements: city NAME and city POPULATION.

ANSWER: The only restriction is that the total number of data elements for all record types within a given logical file cannot exceed 10,000.

5. How many records can exist for a given logical file?

   ANSWER: The total number of records cannot exceed 16,777,215 (2^{24} - 1) for a given logical file.

6. How many distinct peer record types can occur at the same level of file subordination? In the example in Figure 2-1, the CITIES and COUNTIES record types are counted as two peer record types.

   ANSWER: The maximum number of distinct record types that can occur at any given level of file subordination is 10.

7. How many records can exist at any given level of file subordination for a record type such as the CITIES record shown in Figure 2-1?

   ANSWER: The only restriction is that the total number of records for all record types for all levels of file subordination cannot exceed 16,777,215.

8. How many distinct Data Bases can exist at any given computer installation that has implemented the MIRADS System?

ANSWER: There is no maximum number of distinct MIRADS Data Bases that can exist at any given computer installation.

9. Can a subordinate record be linked to more than one superior record at any level such as the DISTRICT record shown in Figure 2-1?

    ANSWER: Currently, a subordinate record can be linked to one, and only one, superior record in the MIRADS logical file structure. The DISTRICT record could not be linked to both the CITIES and COUNTIES records. Network relationships linking subordinate records to two or more peer records will be considered in the future for the MIRADS System.

10. Can null data elements and/or null record types be defined in the logical file structure? In other words, can data elements containing no data values, or record types containing no instances, be provided for at file generation time?

    ANSWER: Data elements which have been defined but have no value can exist in the logical file structure (such as NAME filled in but no value for POPULATION in a CITIES record). Also, a record type may be defined at any level of file subordination but have no occurrences e.g., no values for NAME or POPULATION in the CITIES record.

11. How is the logical organization for a given Data Base chosen?

    ANSWER: The logical file organization of the MIRADS Data Base should match as much as possible the natural relationship of the data elements and records. For example, the data elements COUNTRY NAME and COUNTRY POPULATION could be grouped together to form a COUNTRIES record while the data elements state NAME and state GOVERNOR could be grouped together to form a STATES record, etc. Cities are located in geographical regions called states, and states are located in geographical regions called countries. States are subordinate to countries but superior to cities. This is the natural relationship of the data. Consequently, the logical structure that could be adopted for this particular example is as follows:

```
COUNTRIES
        STATES
            CITIES
```

12. What consideration should be given to the organization of peer records in the MIRADS Data Base?

<u>ANSWER</u>: MIRADS buffer management precludes retrieval of information with one Query command from peer records that are subordinate to the same superior record. In the example in Figure 2-1, the CITIES and COUNTIES records are peer records illustrating this restriction. Information retrieval regarding COUNTRIES, STATES, and CITIES; or COUNTRIES, STATES and COUNTIES is possible. One query regarding information in the NATION, STATES, CITIES, and COUNT'ES records is not possible. If queries of this nature are to be asked, the hierarchical structure should be changed as follows:

```
NATION
    STATES
         COUNTIES
               CITIES
```

2.2.2 <u>Physical File Organization</u>

A physical file is a file of data values reflecting the logical organization of the Data Base. In MIRADS, the natural way of storing hierarchically structured data is used with each group expanded according to its position in the hierarchy. The following questions and answers define the method used by MIRADS to store a Data Base.

1. In what order is the MIRADS Data Base physically stored? Are all values of a data element stored continguously, or is

each record occurrence stored according to its subordinate relationship with other records?

ANSWER: Each record occurrence is stored according to its subordinate relationship, starting with the master record. For example, in Figure 2-1 the country NAME and its PRESIDENT are stored first, followed by the values of state NAME and GOVERNOR for the first state, followed by all the values of city NAME and POPULATION for the first state, followed by the value of county NAME for the first state, followed by all the values of district NAME for the first county, etc., until all county NAMES and corresponding district NAMES have been exhausted for the first state, followed by the values of state NAME and GOVERNOR for the second state, etc. Figure 2-2 depicts the file layout for the logical file structure shown in Figure 2-1.

2. Are physical links or pointers required in the Data Base records to indicate levels of file subordination and peer records to the MIRADS System programs?

ANSWER: No. A data element of one to three characters in length must be located within each physical record and is used for record identification purposes. During the loading

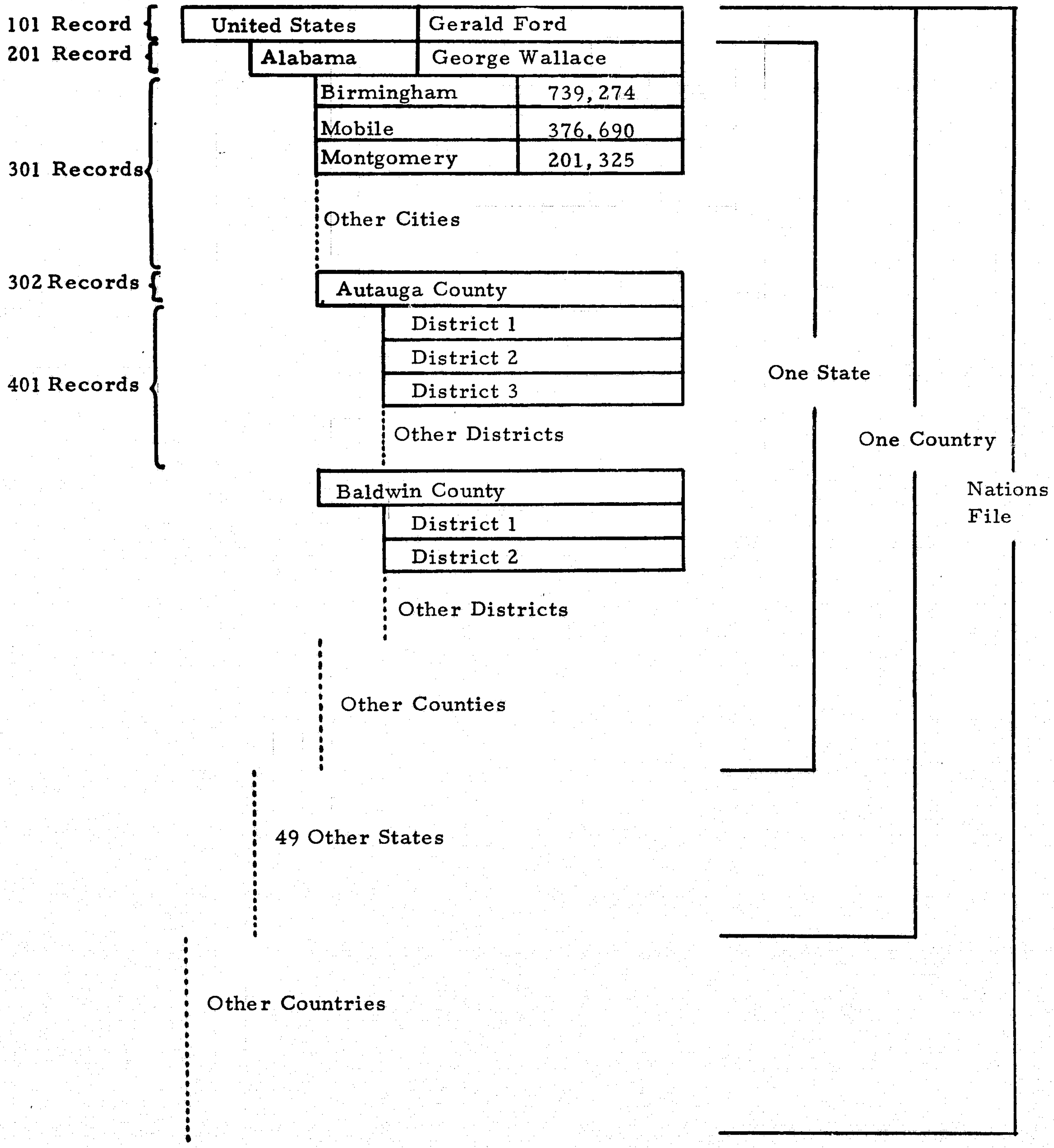


Figure 2-2. File Layout

process for the Data Base, the record identifier is used in conjunction with a Dictionary and the physical order of the MIRADS Data Base records to determine the hierarchy of the file. If the Data Base has no levels of file subordination, and if all records within the Data Base are of the same type, the one-to-three character record identifier described above does not have to be included in the Data Base records.

3. Are variable length records used to create the MIRADS Data Base if more than one record type and length exists in the physical file?

   ANSWER: No. If two or more records of different types and length exist in a file, the shorter records must be padded with spaces to make their length equal to the length of the longest record. This technique is used in MIRADS so that direct addressing of the Data Base can be used rather than the more inefficient methods of indirect addressing or table lookup. Future plans will allow the user to optionally select direct addressing which includes record padding, or some type of indirect addressing which will allow variable length records in the Data Base. Either way, it is a trade-off between mass storage utilization and query response time.

4. What is the maximum record size allowable for a record in the MIRADS Data Base?

   ANSWER: For a Master file with no levels of subordination, the maximum allowable record size is currently set at 5,376 characters or 896 words in length (6 characters per word). Section 3 should be consulted for maximum record sizes when utilizing a Data Base with levels of subordination.

5. Is the MIRADS Data Base written in a blocked or unblocked mode?

   ANSWER: The MIRADS Data Base can be optionally blocked or unblocked at the user's discretion. However, it is strongly recommended for efficiency purposes that the file be blocked as heavily as possible, but not to exceed 1,792 words per block (which is track size for UNIVAC F2 FASTRAND mass storage).

2.2.3 FILE Indexing Scheme

Indexing the MIRADS Data Base provides an efficient means to rapidly retrieve data from the file for immediate use in an on-line environment. The file organization used is the partially inverted Data Base where data is stored in record format form. In addition, certain data elements or fields may be selected as key fields, for which indices will be created. A field is selected as a key field if it is expected to be used rather frequently for retrieval purposes. If experience shows that a selected field is seldom used for retrieval, its index can be deleted. Conversely, if another field requires frequent use, an index can be created for it.

The significant characteristic of an indexed organization is that a table of indices is maintained which points to the records in the Data Base. The index table is searched to find the value of the data element being sought when retrieval is desired. When the value is found, an address is selected from the table entry which points to the specific location of that record in the Data Base. The MIRADS System can then find that position in the file, read the record into memory, and produce the information requested. This method provides for immediate response to on-line queries without resorting to the time-consuming task of sequentially passing, or searching, every record in the Data Base. The following questions and answers provide more insight into the indexing scheme used by MIRADS.

1. In developing the file organization for a MIRADS Data Base, what considerations must be given to indexing?

ANSWER: An analysis should be conducted to determine which data elements would most often be involved in typical user requests for information retrieval. These data elements should probably be indexed if they are involved in the majority of queries requested by the user.

2. In the previous question, it was stated that data elements involved in the majority of queries should probably be indexed. What is meant by "probably"?

ANSWER: Indexing a Data Base is an overhead expense of all Data Base management systems, and is the best technique developed for providing rapid response to on-line queries. The amount of CPU time required during the loading process of the Data Base is directly proportional to the number of indexed data elements selected. Four indexed fields will require approximately twice as much CPU time for loading the Data Base as will two indexed fields. The reason is that every data value for every indexed field must be extracted from the Data Base and sorted to produce an inverted list. It is important to the efficiency of the MIRADS System that as few fields as possible be indexed while still maintaining adequate response to on-line requests for information.

3. If experience shows that one designated indexed field is rarely used for retrieval purpose while another non-indexed field is used frequently, what is involved in deleting the indexed field and adding another?

   ANSWER: First, the Dictionary which describes the data elements in the Data Base to MIRADS must be corrected to indicate those fields which must be indexed. Second, the MIRADS Data Base must be reloaded using the same process as orginially required to load the file.

4. Is the index to the MIRADS Data Base a part of the logical and physical organization of the Data Base?

   ANSWER: No. The MIRADS Data Base and Index files are distinct files stored on an external mass storage device such as a drum or disc. The user should only be concerned with the organization and structure of the MIRADS Data Base. The Index file is automatically generated by MIRADS as a byproduct of the loading process. Its organization and structure are wholly determined by MIRADS and are of no concern to the user. The only responsibility that the user has is the naming or designation of the indexed fields.

5. How are indexed fields specified to MIRADS?

   ANSWER: Associated with every MIRADS Data Base is a Dictionary which describes the data elements or fields in the file to MIRADS. A field which is to be indexed is specified in this Dictionary. (See Section 4 for more information regarding the definition of a MIRADS Dictionary.)

6. What is the minimum and maximum number of indexed fields that can be specified for a MIRADS Data Base?

   ANSWER: The minimum number of indexed fields is 0 and the maximum number is currently set at 200.

7. What is the maximum length, in characters, for an indexed field?

   ANSWER: The maximum length for an indexed field is 24 characters. However, if a data element or field exceeds 24 characters in length, it can still be designated as an indexed field and only the first 24 characters will be indexed.

SECTION 3 - HOW TO CREATE A MIRADS DATA BASE

3.1 MIRADS GENERAL PURPOSE INPUT/OUTPUT ROUTINE

The development of a computerized Data Base for a particular user application inevitably results in the storing of data on some computer storage medium. This medium could be card images, magnetic tape, disc, or drum. The programming language used to assemble the data could be COBOL, FORTRAN, Assembler, PL/1 or some other language. The possibilities for file creation and storage are compounded by considering the different hardware vendors, varying record and block sizes, and differing internal codes used such as Fieldata, ASCII, BCD, EBCDIC, etc.

A generalized Input/Output routine that could read any such file would require complex user instructions, would be inefficient with high overhead, and would be limited in its functions. MIRADS must rapidly and efficiently perform I/O operations on applications files, keeping overhead to a minimum, and providing direct access capability. Consequently, MIRADS provides a common I/O package, named the MIRADS IOPKG, which can be easily accessed by applications programs written in Assembler Language, PL/1, FORTRAN, DOD COBOL, FD COBOL or ASCII COBOL. (See Paragraph 7.2 for MIRADS IOPKG documentation.)

3.2 USING THE MIRADS I/O PACKAGE TO CREATE A DATA BASE

The MIRADS user is required to write his application Master file onto mass storage using the MIRADS IOPKG before it can be used by MIRADS. This file is then referred to within MIRADS as the MIRADS Data Base or Master file and any file written with the MIRADS IOPKG is designated as being in MIRADS format.

The development of a program to write the Master file in MIRADS format is the only programming that is required for the implementation of a Data Base on MIRADS. Most users develop a simple program to read the file in the language that the file was created and to rewrite it using IOPKG. Some users have modified application programs to use the MIRADS IOPKG for all Master file processing to keep from maintaining two different copies of the same data. Two sample ASCII COBOL programs are presented in Paragraphs 3.2.1 and 3.2.2 to illustrate the procedure for creating a MIRADS Data Base from an existing file. The same general procedures used in these programs can be applied to programs written in other languages such as FORTRAN, PL/1, Assembler Language, etc.

3.2.1 Creation of a Single Level Data Base

The program shown in Figure 3-1 illustrates the procedure for creating a MIRADS Data Base with no levels of file subordination from input

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  DBGEN1
REMARKS.        THIS PROGRAM READS AN EXISTING CARD FILE TO
    GENERATE A MIRADS MASTER FILE.
                THE MIRADS MASTER FILE WILL BE STRUCTURED
    WITH NO LEVELS OF FILE SUBORDINATION AND A SINGLE
    RECORD TYPE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  UNIVAC-1108.
OBJECT-COMPUTER.  UNIVAC-1108.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN TO CARD-READER.
DATA DIVISION.
FILE SECTION.
FD CARD-FILE
    LABEL RECORDS ARE OMITTED
    DATA RECORD IS CARD-RECORD.
01  CARD-RECORD                              PICTURE X(84).
WORKING-STORAGE SECTION.
01  MAS-UNIT      VALUE "MAS   "             PICTURE X(12).
01  MAS-REC-SIZE  VALUE 14                   PICTURE H9(10).
01  MAS-BLK-SIZE  VALUE 128                  PICTURE H9(10).
01  MAS-REC-NBR   VALUE 0                    PICTURE H9(10).
01  MAS-RECORD.
    02  MAS-REC   OCCURS 14                  PICTURE X(6).
01  MAS-BUFFER.
    02  MAS-BUF   OCCURS 1807                PICTURE X(6).
PROCEDURE DIVISION.
MASGEN1-OPEN-FILES.
    OPEN INPUT CARD-FILE.
    CALL "OPENS"  USING  MAS-UNIT            MAS-REC-SIZE
                         MAS-BLK-SIZE        MAS-BUFFER.
CARD-READ-IOPKG-WRITE-LOOP.
    READ CARD-FILE INTO MAS-RECORD AT END GO TO CLOSE-FILES.
    ADD 1 TO MAS-REC-NBR.
    CALL "WRITES" USING MAS-UNIT MAS-REC-NBR MAS-RECORD.
    GO TO CARD-READ-IOPKG-WRITE-LOOP.
CLOSE-FILES.
    CLOSE CARD-FILE.
    CALL "CLOSEM" USING MAS-UNIT.
    STOP RUN.
```

Figure 3-1. Data Base Creation Program (DBGEN1)

cards. This program will read 14-word records from the input cards shown in Figure 3-2, and write the records onto the mass storage file, MAS. The output record size, MAS-REC-SIZE, is the same as the input record since no manipulation of the data is required. The blocking factor, MAS-BLK-SIZE, was obtained by calculating the largest number of records that can be stored in one MIRADS file buffer, 1,792 words (1792/14 = 128). The user program is required to furnish the buffer area for IOPKG. The buffer size must be large enough to hold one block of data and must have an additional fifteen words for IOPKG file control information. The buffer for this program, MAS-BUFFER, contains 1,807 words, which was calculated by multiplying the blocking factor, MAS-BLK-SIZE, by the record size, MAS-REC-SIZE, and adding 15 words for a File Control Table (14 x 128 + 15 = 1807).

In the Procedure Division, the call to OPENS initializes IOPKG and establishes the File Control Table for the MIRADS Master file, MAS. Input cards are read directly into the output record since it must be written from this area. The call to WRITES causes the record stored in MAS-RECORD to be placed in the output buffer, MAS-BUFFER, at a relative record location designated by MAS-REC-NBR. This buffer

```
000001JONES       S   C201130[redacted]225 JONES VALLEY      01012027000M06
000002BURNS       W   D320809         1442 NORTH BELAIR     01011023500M04
000005KING        W   L381204         23511 W GEORGIA AVE   04034014400M05
000009ZORNASKI    P   U360424         4244 JEAN ROAD        05043015000F01
000011THOMPSON    J   K450617         7504 DOWNING DRIVE    01011009600M01
```

Figure 3-2. DBGEN1 Input Data Cards

will be automatically written to mass storage by IOPKG when the buffer is filled. The call to CLOSEM is required to properly close the file by purging the file buffer and writing a software end-of-file after the last record written. Programs that have ample core available can realize a reduction of approximately 20 percent in elapsed time by using the IOPKG double buffering capability. (See Paragraph 7.2 for detailed documentation of IOPKG.)

This example is representative of the simplest and most common type of application for generation of a MIRADS Master file. There is only one type of data record, so there is no need to get involved with hierarchical file structuring of the Master file. All data elements within the existing file are to be used in the MIRADS Data Base, and all are in proper format for MIRADS use; consequently, the MIRADS Data Base record can be written directly as it is read from cards without any data manipulation.

3.2.2 Creation of a Multi-Level Data Base

Many applications will fall into the above category, but the Master File Generator program can become as complex as required by the user. The program shown in Figure 3-3 illustrates the generation of a MIRADS Master file with a hierarchical structure and minor manipulation

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   DBGEN2.
REMARKS.
          THIS PROGRAM GENERATES A MIRADS MASTER FILE FROM AN
     EXISTING CARD FILE.  THE MIRADS MASTER FILE WILL BE
     STRUCTURED WITH TWO LEVELS OF FILE SUBORDINATION.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNIVAC-1108.
OBJECT-COMPUTER. UNIVAC-1108.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT CARD-FILE ASSIGN TO CARD-READER.
DATA DIVISION.
FILE SECTION.
FD CARD-FILE
     LABEL RECORDS ARE OMITTED
     DATA RECORDS ARE COUNTRY-CARD STATE-OR-COUNTY-CARD.
01   COUNTRY CARD.
     02  FILLER                                       PICTURE X(4).
     02  CARD-TYPE                                    PICTURE X.
     02  FILLER                                       PICTURE X(3).
     02  COUNTRY-CARD-DATA.
         03  COUNTRY-AND-PRESIDENT                    PICTURE X(48).
         03  POPULATION-COUNTRY                       PICTURE X(10).
         03  AREA-COUNTRY                             PICTURE X(8).
         03  FILLER                                   PICTURE X(10).
     02  FILLER
01   STATE-OR-COUNTY-CARD.
     02  FILLER                                       PICTURE X(8).
     02  STATE-OR-COUNTY-DATA.
         03  STATE-OR-COUNTY                          PICTURE X(14).
         03  CAPITAL-OR-SEAT                          PICTURE X(16).
         03  POPULATION-STATE-COUNTY                  PICTURE X(8).
         03  AREA-STATE-COUNTY                        PICTURE X(6).
         03  STATE-RANK-IN-POP                        PICTURE X(2).
         03  STATE-RANK-IN-AREA                       PICTURE X(2).
         03  STATE-GOVERNOR                           PICTURE X(24).
```

Figure 3-3. Data Base Generation Program (DBGEN2)

```
WORKING-STORAGE SECTION.
01   MAS-UNIT        VALUE "MAS       "        PICTURE X(12).
01   MAS-REC-SIZE    VALUE 13                  PICTURE H9(10).
01   MAS-BLK-SIZE    VALUE 137                 PICTURE H9(10).
01   MAS-REC-NBR     VALUE 0                   PICTURE H9(10).
01   MAS-RECORD.
     02  MAS-REC-TYPE                          PICTURE 9(3).
     02  FILLER      VALUE SPACES              PICTURE X(3).
     02  MAS-REC-DATA.
         03  MAS-DATA OCCURS 12 TIMES          PICTURE H9(10).
01   MAS-BUFFER.
         03  MAS-BUF  OCCURS 1796 TIMES        PICTURE H9(10).
PROCEDURE DIVISION.
MASGEN-OPEN-FILES.
     OPEN INPUT CARD-FILE.
     CALL "OPENS" USING  MAS-UNIT              MAS-REC-SIZE
                         MAS-BLK-SIZE          MAS-BUFFER.
READ-CARD-FILE.
     READ CARD-FILE AT END GO TO CLOSE-FILES.
     IF CARD-TYPE = "A" MOVE 101 TO MAS-REC-TYPE
       PERFORM COUNTRY-ZERO-FILL
       MOVE COUNTRY-CARD-DATA TO MAS-REC-DATA
       GO TO WRITE-MAS-RECORD.
     IF CARD-TYPE = "B" MOVE 201 TO MAS-REC-TYPE
       PERFORM STATE-ZERO-FILL THRU COUNTY-FILL
       MOVE STATE-OR-COUNTY-DATA TO MAS-REC-DATA
       GO TO WRITE-MAS-RECORD.
     IF CARD-TYPE = "C" MOVE 301 TO MAS-REC-TYPE
       PERFORM COUNTY-FILL
       MOVE STATE-OR-COUNTY-DATA TO MAS-REC-DATA
       GO TO WRITE-MAS-RECORD.
```

Figure 3-3. Data Base Generation Program (DBGEN2) (Continued)

```
INVALID-INPUT-CARD.
    DISPLAY "INVALID REC TYPE " STATE-OR-COUNTY-CARD.
    GO TO READ-CARD-FILE.
COUNTRY-ZERO-FILL.
    IF POPULATION-COUNTRY IS NOT EQUAL TO SPACES
      EXAMINE POPULATION-COUNTRY REPLACING LEADING
      SPACES BY ZERO.
    IF AREA-COUNTRY IS NOT EQUAL TO SPACES
      EXAMINE AREA-COUNTRY REPLACING LEADING
      SPACES BY ZERO.
STATE-ZERO-FILL.
    IF STATE-RANK-IN-POP IS NOT EQUAL TO SPACES
      EXAMINE STATE-RANK-IN-POP REPLACING LEADING
      SPACES BY ZERO.
    IF STATE-RANK-IN-AREA IS NOT EQUAL TO SPACES
      EXAMINE STATE-RANK-IN-AREA REPLACING LEADING
      SPACES BY ZERO.
COUNTY-FILL.
    IF POPULATION-STATE-COUNTY IS NOT EQUAL TO SPACES
      EXAMINE POPULATION-STATE-COUNTY
              REPLACING LEADING SPACES BY ZERO.
    IF AREA-STATE-COUNTY IS NOT EQUAL TO SPACES
      EXAMINE AREA-STATE-COUNTY REPLACING LEADING
      SPACES BY ZERO.
WRITE-MAS-RECORD.
    ADD 1 TO MAS-REC-NBR.
    CALL "WRITES" USING MAS-UNIT MAS-REC-NBR MAS-RECORD.
    GO TO READ-CARD-FILE.
CLOSE-FILES.
    CLOSE CARD-FILE.
    CALL "CLOSEM" USING MAS-UNIT.
    STOP RUN.
```

Figure 3-3. Data Base Generation Program (DBGEN2) (Continued)

of the data. This program reads the card file of Figure 3-4 and generates a MIRADS Data Base with two levels of subordination with the following structure:

```
COUNTRY
    STATE
        COUNTY
```

The input card file is in sequence by card columns 1 through 8, with columns 1-2 identifying all records within a particular COUNTRY, columns 3-4 identifying all records within a particular STATE, and columns 6-8 identifying all records within a particular COUNTY. In this example, there is only one COUNTRY, so all records containing 01 in columns 3-4 are within Alabama. The COUNTY records are the lowest level of file subordination, so each county has a unique code in columns 6-8. Column 5 contains CARD-TYPE, which is used to identify the different record types, i. e., all records containing an A in column 5 are COUNTRY records, all records containing a B in column 5 are STATE records, and all records containg a C in column 5 are COUNTY records.

All hierarchically structured files must be in the proper logical sequence before being written to the MIRADS Data Base. If the card file in the example was out of sequence, a sort could be embedded in the Data Base Generator program to sequence the file before writing the

```
01        UNITED STATES OF AMERICAFORD, GERALD R          203235298 36086723
0101B    ALABAMA           MONTGOMERY        3444165 516902129WALLACE, GEORGE C
0101C001AUTAUGA            PRATTVILLE          24460    599
0101C002BALDWIN            BAYMINETTE          59382   1578
0101C003BARBOUR            CLAYTON             22543    891
0101C037JEFFERSON          BIRMINGHAM         644991   1115
0101C045MADISON            HUNTSVILLE         186540    803
0103B    CALIFORNIA        SACRAMENTO       19953134158693010 3BROWN, EDMUND G JR
0103CC01ALAMEDA            OAKLAND           1073184    733
0103C01SL0S  ANGELES       LOS ANGELES       7036887   4069
0151B     WYOMING          CHEYENNE           332416 979145009HATHAWAY, STANLEY K
0151C023WESTON             NEW CASTLE           6307   2407
```

Figure 3-4. DBGEN2 Input Cards

MIRADS Data Base. MIRADS must be able to identify different types of records within hierarchical files or within complex files with more than one type of record. This is accomplished by including a one-to-three-character record identifier as part of each record. This identifier is used by MIRADS to identify the type of record and level of subordination of the record. (See Record Identifier Card of the Dictionary in Paragraph 4.2.3.)

In the sample program, when an A card type is recognized, the value 101 is moved into columns 1 through 3 of the MIRADS Master record to identify it as a COUNTRY record. The A could be used as well, but most MIRADS users find it easier to use the 3-digit code which identifies the level of file subordination and type of record within that level. A value of 201 is moved into the MIRADS Master record to identify STATE records and a value of 301 is used for COUNTY records.

The population, area, and rank fields are zero-filled to the left by the program. This is required so that fields used for searching, printing, computing, and sorting will be uniform in content for each data type. Similarly, other user Data Base Generator programs may be required to adjust and align other types of fields to conform to MIRADS requirements for uniformity of fields. (See Paragraph 3.3.1 for allowable Data types and field alignments.) The data from all three record types is moved to the same output record area, MAS-RECORD, to be written. The STATE record is the largest, containing 12 words of data plus one additional word to contain Record Type. IOPKG does not presently support variable length records, so the smaller COUNTRY and COUNTY records are padded with spaces to be equal in length (13 words) to the STATE Record. The value of record size, in words, for the MIRADS Data Base (MAS-REC-SIZE) must be the size of the largest record, 13 words. The blocking factor, MAS-BLK-SIZE, calculates to a value of 137 (1792/13 = 137), and the buffer for IOPKG, MAS-BUFFER, must be at least 1796 words long (13 x 137 + 15 = 1796). Procedures for calculating the blocking factor and buffer areas are discussed in Paragraph 3. 3. 3.

3. 3 THINGS TO KNOW BEFORE CREATING A DATA BASE

Before a user can write a MIRADS Data Base generation program he must have specific knowledge about the types of data supported by MIRADS, buffer management techniques, and limitations imposed by MIRADS that affect the contents and structure of the MIRADS Master file. The remaining portion of Section 3 is devoted to supplying answers that the analyst might need to know before developing the program to write his MIRADS Master file.

3. 3. 1 Data Types Supported by MIRADS

MIRADS is written in ASCII COBOL with the exception of some highly specialized I/O and data manipulation routines. Although ASCII COBOL is the Compiler, MIRADS executes in Fieldata mode and does not support the overpunch sign feature of the FD COBOL or ASCII COBOL compilers. All signed numeric fields, with the exception of binary integer and floating point fields, must have the sign present and the sign takes one character position at the high order position of the field (+001234). The sign character and leading zeros on Fieldata numeric fields are essential in forming proper criteria for searching the Master file.

All fields can be searched for the presence or absence of data. To be uniform for all types of fields, the presence of any non-blank Fieldata character will indicate the presence of data within the field. Any field filled with Fieldata spaces will then be considered to be absent of data. The binary integer and floating point data fields could conceivably contain valid data values of all Fieldata spaces, but the presence and absence of fields is a search technique and does not conflict with use of the field for computing and sorting. Following are the nine data types currently supported by MIRADS and some of the important field characteristics.

1. <u>Alphabetic</u>

Fieldata Characters:	6 bits per character
Maximum Field Length:	48 characters, or 132 characters if the field is used as a print only field

Allowable Characters:	Any letter of the alphabet or a space

Field should be left-justified and space-filled to the right.

2. Alphanumeric

Fieldata Characters:	6 bits per character
Maximum Field Length:	48 characters, or 132 characters if the field is used as a print only field
Allowable Characters:	Any character from the computer's character set

Field should be left-justified and space-filled to the right.

3. Fieldata Numeric, Unsigned, Assumed Decimal

Fieldata Characters:	6 bits per character
Maximum Field Length:	18 characters
Maximum Number of Decimal Digits:	8
Allowable Characters:	0 through 9

Field must be right-justified and zero-filled to the left.

4. Fieldata Numeric, Signed, Assumed Decimal

Fieldata Characters:	6 bits per character
Maximum Field Length:	18 characters which includes a sign character
Maximum Number of Decimal Digits:	8

Allowable Characters: 0 through 9 or + or -

Field must be right-justified and zero-filled to the left. The high order character position must contain a plus or minus sign. Positive fields must have a sign present for proper searching and sorting within MIRADS.

5. <u>Fieldata Numeric, Unsigned, Decimal Present</u>

Fieldata Characters:	6 bits per character
Maximum Field Length:	18 characters and which includes a decimal
Maximum Number of Decimal Digits:	8
Allowable Characters:	0 through 9 or .

Field must be right-justified and zero-filled to the left. The decimal point must be present within the field.

6. <u>Fieldata Numeric, Signed, Decimal Present</u>

Fieldata Characters:	6 bits per character
Maximum Field Length:	18 digits and which include a sign and a decimal
Maximum Number of Decimal Digits:	8
Allowable Characters:	0 through 9 or . or + or -

Field must be right-justified and zero-filled to the left. The high order character position must contain a plus or minus sign. The decimal point must be present within the field.

Positive fields must have a sign present for proper searching and sorting within MIRADS.

7. Binary Integer

FORTRAN:	Single precision Integer data
COBOL:	PICTURE SH9(10) or H9(10)

Field must be 36 bits (6 character positions) in length. Field should be word aligned, but word alignment is not required by MIRADS. The capability of searching and printing up to 10 significant numeric digits ($\pm 2^{35} -1$) exists.

8. Single Precision Floating Point

FORTRAN:	Single precision Floating Point field
ASCII COBOL:	USAGE IS COMPUTATION-1

Field must be 36 bits (6 character positions) in length. Field should be word aligned, but word alignment is not required by MIRADS. The capability of searching up to nine significant numeric digits and printing up to eight significant numeric digits exists.

9. Double Precision Floating Point

FORTRAN:	Double precision floating point field
ASCII COBOL:	USAGE IS COMPUTATIONAL-2

Field must be 72 bits (12 character positions) in length. Field should be word aligned, but word alignment is not required by MIRADS. The capability of searching up to 18 significant numeric digits and printing up to 8 significant numeric digits exists.

3.3.2 Field Content and Search Criteria

The way a field is to be used by MIRADS determines how this field should be stored in the MIRADS Data Base. The fields to be used for computation have to be numeric fields. Whether these fields should be binary, floating point, or numeric Fieldata must be determined by the user, and will normally be dictated by the type of field in the application Master file.

Fields containing information such as calendar dates require manipulation within the Data Base generator program. Dates with the format MM-DD-YY are good for printing, but make searching for a range of dates very difficult. There are three ways to solve this and similar problems. The field could be stored in the MIRADS Data Base in the format YYMMDD for ease of searching. The date in this format does not look as good on the printed output and may not be acceptable for some formal reports. A second solution would be to store the date in both formats as two separate fields. One field could be used for searching and the other for printing. The third alternative would be to store the date in MM-DD-YY format and define the date as eight alphanumeric characters. (See Section 4 for detailed discussion of the

Dictionary.) The year and month can be defined as two-digit fields and can be used as individual fields in searching the Data Base, while the eight-character field would be used for printing. The third approach is the most widely used technique because it permits flexibility without requiring additional storage space in the record.

Consideration must also be given to large alphabetic and alphanumeric fields. Fields larger than 48 characters in length are normally used only for printing purposes since the search criteria in the MIRADS Query command is limited to 48 characters. However, there are three types of special searches that can be used on fields up to 132 characters in length. The keyword, keyphrase, and character searches (see Paragraph 4.2.4) permit the user to search for up to 48 characters of specific data within any defined Fieldata field. The maximum allowable field size is 132 characters. Fields larger than this can be contained within the MIRADS Master file, but cannot be defined to MIRADS through the use of the Dictionary.

3.3.3 Using MIRADS Buffers Efficiently

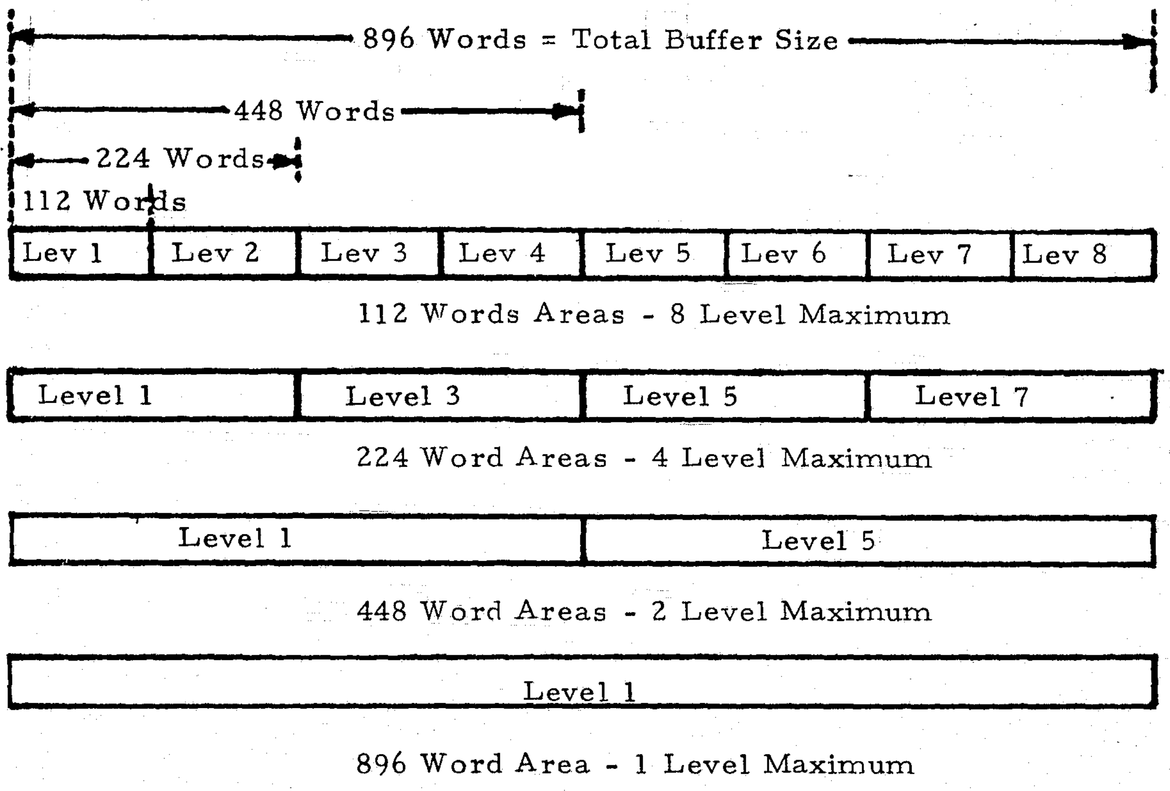
The buffers in the MIRADS System programs which process the user's Master file are maintained at a reasonably large size to maintain input/output efficiency, but at the same time not so large as to require excessive amounts of core in the computer. The Master file buffer is 1,792 words long, and user Master file blocks larger than this cannot be handled by MIRADS.

User records should be blocked as close to 1,792 words as possible for efficient processing of the Master file. To calculate the block size or blocking factor, the record size in words is divided into 1,792, and this block size is used in the call to the OPENS entry point of the MIRADS IOPKG when creating the Master file. The user program that creates the Master file has to supply a file buffer large enough to hold one block of data plus an extra 15 words for a File Control Table. To calculate the minimum buffer area required for the file, the block size or blocking factor is multiplied by the record size and 15 words are added. The formulas for calculating the blocking factor and the buffer size are:

```
  BLOCK SIZE = 1792 / RECORD SIZE
BUFFER SIZE = BLOCK SIZE x RECORD SIZE + 15
```

The user is not required to block to full capacity, but it is done to minimize record access time when processing the user's Master file.

The MIRADS System programs provide a record buffer of 896 words for processing all the various record types in a user's Master file. The record buffer permits a maximum rècord size of 896 words; however, this is only true for a file with no levels of subordination. For hierarchical files, the record buffer can be considered as eight concurrent 112-word buffer areas. Level-1 type records start loading into the first area, Level-2 types into the second, Level-3 types into the third, etc. With this type of loading, multi-level Master file record sizes are limited to 112 words. This limit can be varied by using a unique MIRADS structuring technique. The limit on record size can be increased to 224 words by defining only Levels 1, 3, 5, and 7 to MIRADS. (See Paragraphs 4.2.1 and 4.2.2.) The absence of Levels 2, 4, 6, and 8 permits data from the odd levels to extend into their buffer areas. Similarly, the limit can be increased to 448 words by defining only Levels 1 and 5, and increased to 896 words by defining only Level 1 record types. Figure 3-5 illustrates the relationship of record size and levels of file subordination. This technique of loading into a specific buffer area based on the level of file subordination precludes two or more peer records being loaded at the same time. Consequently, a single Query command requesting information from two or more peer records is not possible within MIRADS. This restraint, as well as the restraints on Master file record sizes, should be taken into consideration by the user when creating a MIRADS Data Base.


896 Words = Total Buffer Size
448 Words
224 Words
112 Words
Lev 1
Lev 2
Lev 3
Lev 4
Lev 5
Lev 6
Lev 7
Lev 8
112 Words Areas - 8 Level Maximum
Level 1
Level 3
Level 5
Level 7
224 Word Areas - 4 Level Maximum
Level 1
Level 5
448 Word Areas - 2 Level Maximum
Level 1
896 Word Area - 1 Level Maximum

Levels of File Subordination	Maximum Record Length (In Words)	Permissible Levels of Records
0	896	1
1	448	1, 5
2	224	1, 3, 5
3	224	1, 3, 5, 7
4	112	1, 2, 3, 4, 5
5	112	1, 2, 3, 4, 5, 6
6	112	1, 2, 3, 4, 5, 6, 7
7	112	1, 2, 3, 4, 5, 6, 7, 8

Figure 3-5. MIRADS Record Buffer

SECTION 4 - HOW TO DEFINE A DATA BASE TO MIRADS

4.1 INTRODUCTION

Data definition is the process by which the user describes a particular Data Base to MIRADS. As part of the data definition process, symbolic names of the data elements in the Data Base are defined along with their logical structure and physical properties. Information is specified to define the location, size, type of data, etc., of a data element, and its relationship with other data elements in the Data Base.

The user must employ a data definition language called a Dictionary to define a Data Base. The Dictionary is the cornerstore of the MIRADS System, providing a centralized definition of data, thereby allowing the separation of data from the computer programs which process the data. The Dictionary provides an interface to a set of generalized programs so that these programs can work on several different Data Bases without regard to their content. This set of generalized programs is the MIRADS System and the inventory of all the capabilities it provides. Section 4 describes in detail the data definition language, or Dictionary, used to describe a Data Base to MIRADS.

4.2 DICTIONARY INPUT CARDS

MIRADS provides an update edit program which creates the MIRADS Dictionary from card input. The program edits the input cards to verify that each input parameter has been entered correctly, and produces an output report or listing summarizing the parameter information. If errors exist in the input cards, appropriate diagnostic messages are produced on an error listing describing the cause for the error. Additional information on the use of the Dictionary generation program is given in Section 5.

The following paragraphs describe in detail the data cards that must be input to the Dictionary generation program. A maximum of five input cards may be used for Dictionary generation. They are the Filename, Password, Record Identifier, Field Definition, and Table Lookup cards. However, the Password and Table Lookup cards are optional, making only three input cards necessary for Dictionary generation.

There are three types of input transactions to the Dictionary. They are Insert, Delete, and Modify. Each input card must indicate the type of transaction as well as card type; that is, whether it is a Filename card or one of the four other cards.

4.2.1 Filename Card

The function of the Filename card is to name the MIRADS Data Base that is being described, and to provide other pertinent information relating to the definition of the Data Base. This information includes the approximate number of records in the Data Base, the size of the records in the Data Base, the number of records per block, and the number of levels of file subordination contained within the Data Base.

Each field of the Filename card, as shown in Figure 4-1, is explained along with a description of the contents of each field. Each MIRADS Data Base requires only one Filename card.

FILENAME CARD

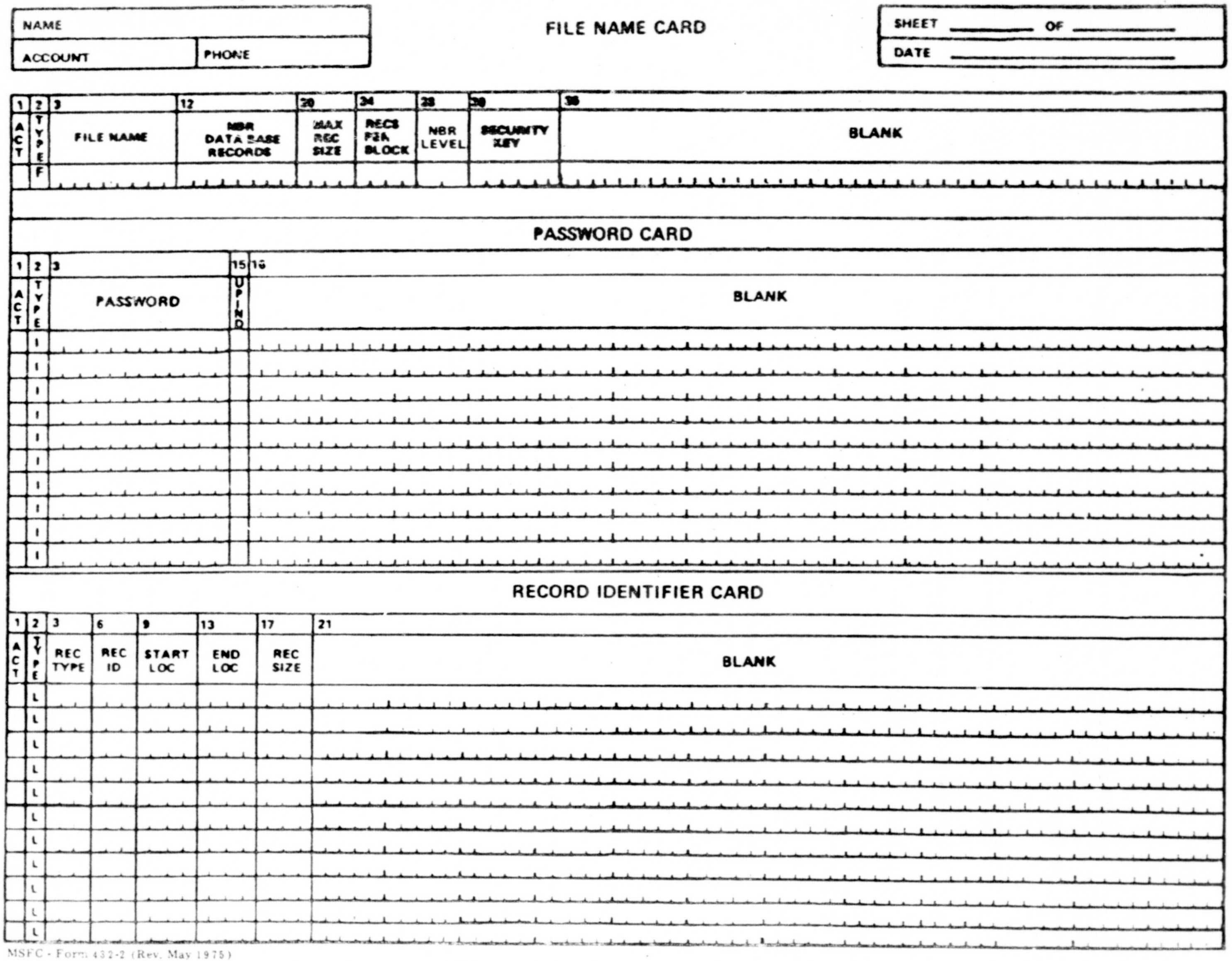
ACT - Card column 1. Action code indicating one of three possible types of Dictionary input transactions. The action code must be I for Insert, D for Delete, or M for Modify. The Insert action requires that all fields of the Filename card be completed as described below. The Delete action requires that the ACT, TYPE, and FILENAME fields be completed. The Modify action requires completion of the ACT, TYPE, and FILENAME fields as well as the fields to be altered. Only the values of those fields specified on the Filename card are altered by the Modify action.

TYPE - Card column 2. Card type indicating the type of Dictionary input card. A constant value of F must be used to denote Filename card.

FILENAME - Card columns 3-11. Unique name used to identify the Data Base described by this Dictionary using one to nine alphanumeric characters left-justified.

NBR DATABASE RECORDS - Card columns 12-19. Approximate number of records in the Data Base described by this Dictionary, and is right-justified. A maximum estimate within 10 percent of the actual number of records expected in the Data Base should be used. This value is used in establishing parameters which significantly affect the efficiency of the MIRADS load programs. If the estimate is 10 percent less than the actual number of records, the MIRADS load programs could abort with a sort B5 error.

MAX REC SIZE - Card columns 20-23. Maximum record size designating the number of words in the largest record in the Data Base described by this Dictionary, and is right-justified. Since MIRADS



NAME	
ACCOUNT	PHONE

FILE NAME CARD

SHEET ______ OF ______
DATE ______

1	2	3	12	20	24	28	30	36
ACT	TYPE	FILE NAME	NBR DATA BASE RECORDS	MAX REC SIZE	RECS PER BLOCK	NBR LEVEL	SECURITY KEY	BLANK
	F							

PASSWORD CARD

1	2	3	15	16
ACT	TYPE	PASSWORD	UPIND	BLANK
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			

RECORD IDENTIFIER CARD

1	2	3	6	9	13	17	21
ACT	TYPE	REC TYPE	REC ID	START LOC	END LOC	REC SIZE	BLANK
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						

MSFC - Form 432-2 (Rev. May 1975)

Figure 4-1. File Name Card

does not support variable length records per se, records of varying sizes must be padded or space-filled to produce uniform record sizes.

RECS PER BLOCK - Card columns 24-27. The number of records per block in the Data Base, and is right-justified. The maximum value allowed is equal to 1,792 divided by the MAX REC SIZE.

NBR LEVELS - Card columns 28-29. The number of the highest level of file subordination referenced within the Data Base described by this Dictionary. The highest level is determined by the Dictionary generation program and overrides any value specified by the user. Consequently, this field may be left blank. The value must be in the range 1-8, and right-justified. See REC TYPE of the Record Identifier card for additional information regarding the contents of this field.

SECURITY KEY - Card columns 30-35. Security key is an optional field used to control access to the data element level for the Data Base described by this Dictionary. Further information regarding the use of this field is described under separate cover and can be provided on a need-to-know basis. Security key should be left blank if security is not required for the MIRADS Data Base.

BLANK - Card columns 36-80 must be all blank.

4.2.2 Password Card

The function of the Password card is to define a Password which may be used to gain access to a MIRADS Data Base. Fields of the card include the card type, transaction type, update indicator, and security information. The Data Base may or may not be updated, depending on the update indicator of the Password being used. The Password card is an optional input card which may be used one or more times, depending on the number of Passwords desired for a given Data Base.

Each field of the Password card, as shown in Figure 4-1, will be explained along with a description of the contents of the fields.

PASSWORD CARD

ACT - Card column 1. Action code indicating one of three possible types of Dictionary input transactions. The action code must be I for Insert, D for Delete, or M for Modify. The Insert action requires that all fields of the Password card be completed as described below. The Delete action requires that the ACT, TYPE, and PASSWORD fields be completed. The Modify action requires completion of the ACT, TYPE, and PASSWORD fields as well as the fields to be altered. Only the values of those fields specified on the Password card are altered by the Modify action.

TYPE - Card column 2. Card type indicating the type of Dictionary input card. A constant value of I must be used to denote Password card.

PASSWORD - Card columns 3-14. Password is a unique name used at Query execution time to control access at the file level for the Data Base described by this Dictionary. It is 1-to-12 alphanumeric characters in length and must be left-justified.

UP IND - Card column 15. Update indicator shows whether or not the MIRADS UPDATE command may be used with this Password while using the Data Base described by this Dictionary. A value of 'Y' must be used to indicate that updating is possible, otherwise a value of 'N' or space indicates a non-updatable password.

SECURITY KEYS - Card columns 16-65. Security keys is an optional field used to control access to the data element level for the Data Base described by this Dictionary. Further information regarding the use of this field is described under separate cover and can be provided on a need-to-know basis. Security keys should be left blank if security is not required for the MIRADS Data Base. Note, the Password card may be used to control access at the file level for a Data Base without the use of security keys to control access to a Data Base at the data element level.

BLANK - Card columns 66-80 must be all blank.

4.2.3 Record Identifier Card

The function of the Record Identifier card is to define the various types

of records contained within a MIRADS Data Base. Descriptive information includes an identifier for each record type, the relative start and end location in a record where this identifier can be found, and the size of the record which is being described.

Each field of the Record Identifier card, as shown in Figure 4-1, is explained as well as the contents of the fields. One Record Identifier card is required for each distinct record type contained within the Data Base.

RECORD IDENTIFIER CARD

ACT - Card column 1. Action code indicating one of three possible types of Dictionary input transactions. The action code must be I for Insert, D for Delete, or M for Modify. The Insert action requires that all fields of the Record Identifier card be completed as described below. The Delete action requires that the ACT, TYPE, and REC TYPE fields be completed. The Modify action requires completion of the ACT, TYPE, and REC-TYPE fields as well as the fields to be altered. Only the values of those fields specified on the Record Identifier card are altered by the Modify action.

TYPE - Card column 2. Card type indicating the type of Dictionary input card. A constant value of L must be used to denote Record Identifier card.

REC TYPE - Card columns 3-5. Record type is used to indicate the kind of record being described in the Data Base, and is broken into two parts: The first part, composed of one character, indicates the level of file subordination for the record and must have a value ranging from 1-8. The base level of file subordination is designated as 1, the first level of subordination is designated as 2, etc. The second part, composed of two characters, is used to distinguish one record format from another at the given level of file subordination, and must have a value ranging from 01-10. For example, a Data Base with two levels of file subordination contains CITIES and COUNTIES record types at the third level. The REC TYPE for the CITIES and COUNTIES records would be 301 and 302, respectively. For a Data Base with no levels of file subordination and only one record type, a value of 101 for REC TYPE is suggested. Example:

REC TYPE	RECORDS
101	COUNTRY
201	STATES
301	CITIES
302	COUNTIES

REC ID - Card columns 6-8. Record identifier is a one-to-three-character alphanumeric value physically located in the Data Base records. This value is used to distinguish one record format from another and must be left-justified. For example, a Data Base containing CITIES and COUNTIES records could have a value of 'A' in character one of each CITIES record while the COUNTIES records could have a value of 'B' in character one of each record. The values, 'A' and 'B', could then be used to identify one record format from another. For a file with no levels of subordination and only one record type, REC ID should be '///'.

START LOC - Card columns 9-12. Start location indicates the starting position in the Data Base records where the record identifier (REC ID) described above may be found. The first character in a record is designated as start location 1, the second character is 2, etc. This field may be left blank for a Data Base with no levels of file subordination and only one record type; otherwise, it must be right-justified.

END LOC - Card columns 13-16. End location indicates the ending position in the Data Base records where the record identifier (REC ID) is described above may be found. The first character in a record is designated as end location 1, the second character is 2, etc. This field may be left blank for a Data Base with no levels of file subordination and only one record type; otherwise, it must be right-justified.

REC SIZE - Card columns 17-20. Record size indicates the size in words of the record type in the Data Base being described by this Record Identifier card. This field must have the same value as MAX REC SIZE on the Filename card, and must be right-justified.

BLANK - Card column 21-80 must be all blank.

4.2.4 Field Definition Card 1

The function of the Field Definition Card 1 is to name and define the data elements or data fields of the Data Base to the MIRADS System. Typical information provided for each data field includes the start and end location of the field, the record type to which the field belongs, the type of search which can be performed on the field, and whether or not a field can be updated. Each field of Field Definition 1 card, as shown in Figure 4-2, is explained as well as the contents of the fields. One Field Definition Card 1 is required for each data element of the Data Base being described.

FIELD DEFINITION CARD 1

ACT - Card column 1. Action code indicating one of three possible types of Dictionary input transactions. The action code must be I for Insert, D for Delete, and M for Modify. The Insert action requires that all fields of Field Definition Card 1 be completed as described below. The Delete action requires that the ACT, TYPE, FIELD NBR, and NBR fields be completed. The Modify action requires completion of the ACT, TYPE, FIELD NBR, and NBR fields as well as the fields to be altered. Only the values of those fields specified on the Field Definition card are altered by the Modify action.

TYPE - Card column 2. Card type indicating the type of Dictionary input card. A constant value of M must be used to denote Field Definition card.

FIELD NBR - Card columns 3-6. Field number is a unique number used to identify the data element being described by this Field Definition card. The value for field number must be right-justified and in the range from 0001 to 9999.

NBR - Card column 7. Constant value of 1.

NAME		
ACCOUNT	PHONE	

FIELD DEFINITION CARD 1

SHEET	OF
DATE	

1	2	3 6	7	8	20	56	59	63	67	69	71	72	73	75	76	77	80
ACT	TYPE	FIELD NBR	NBR	FIELD NAME	REPORT TITLE	REC TYPE	START LOC	END LOC	NBR OCC	SRCH TYPE	INDEX	UP IND	DATA TYPE	DECS IN	DECS OUT	TLU TABLE NBR	GTLU
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														

MSFC - Form 432-4 (Rev. May 1975)

Figure 4-2. Field Definition Card 1

FIELD NAME - Card columns 8-19. Field name is a unique name assigned to the data element being described by this Field Definition card. Characters allowed to form this name are A through Z, 0 through 9, and dash (-). Field name must not be a MIRADS reserved word or symbol (see MIRADS Reserved Words and Symbols in Table 4-1), and must be left-justified.

REPORT TITLE - Card columns 20-55. Report title indicates the title that will be printed on output reports for the Field Name described above. Report title is an alphanumeric field, should be left-justified, and may have up to five intervening spaces.

REC TYPE - Card columns 56-58. Record type on the Field Definition card must correspond to one of the record types (REC TYPE) previously defined on the Record Identifier card. It indicates that the data element being described on this card belongs to, or is a member of, one of the previously defined record types.

START LOC - Card columns 59-62. Start location indicates the starting position in the Data Base record where the data element being described on this card may be found. The first character in a record is designated as start location 1, the second character is 2, etc. This field must be right-justified.

END LOC - Card columns 63-66. End location indicates the ending position in the Data Base record where the data element being described on this card may be found. The first character in a record is designated as end location 1, the second character is 2, etc. This field must be right-justified.

NBR OCC - Card columns 67-68. Number of occurrences indicates how many times that the data element being described repeats. This capability is not presently operational in MIRADS and the field should be left blank.

SRCH TYPE - Card columns 69-70. Search type indicates one of four types of searches which can be used to satisfy MIRADS inquiries about a data element. The values for this field are:

RG for Regular Search or
CH for Character Search or
KW for Keyword Search or
KP for Keyphrase Search or
Blank defaults to Regular search.

Table 4-1. MIRADS Reserved Words and Symbols

Reserved Words		
	GREATER-EQUAL	P
A	GT	PRESENT
ADD	GREATER	PRINT
ALL	G	PAGENR
ASCENDING	GROUP	PPOS
BREAK	I	Q
BREAK-COUNT	KEY	QUERY
BRK	KP	RG
BRK-CNT	KW	R
C	LIM	RJ
CH	LIMIT	RJT
CHECK	LESS-EQUAL	RUN
COMPUTE	LE	S
COUNT	LT	SORT
D	LESS	SUM
DELETE	L	SM
DESCENDING	LOOKUP	SP
DEF	LJ	SPACE
DEFINE	LJT	SPLIT
DISPLAY	LIST	SAVE
DO	NP	SAVEC
DRUM	NOT	STANDARD
EDIT	NOT-PRESENT	TLU
EQ	NE	UNEQUAL
EQUAL	NONE	U
EQUALS	NEW	UPDATE
F	OR	
FORMAT		
FULL		
GE		

SYMBOLS

$ = / * ( ) +

When an inquiry involving the data element described on this card is made, this field is used to determine the type of search required to satisfy the inquiry. However, type of search as defined here can be overridden at inquiry time through the use of the MIRADS QUERY command.

When determining the type of search to be used, the length of a data element (START LOC - END LOC + 1) should not exceed 48 characters for a Regular search, or 132 characters for a Character, Keyword, or Keyphrase search. More information about the types of searches may be found in Paragraph 3.3.2 of the MIRADS USER'S MANUAL.

INDEX - Card column 71. Index is used to indicate an indexed field and provides an efficient means to rapidly retrieve data from the Data Base. A value of Y indicates an indexed field while leaving it blank indicates a non-indexed field. See Paragraph 2.2.3 for more information regarding the use of indexed fields.

UP IND - Card column 72. Update indicator shows whether the data element described on this card can be updated or not. A value of Y indicates the data element may be updated while leaving it blank indicates a non-updatable data element.

DATA TYPE - Card columns 73-74. Data type indicates the kind of data contained in the data element described by this card. The data types supported by MIRADS are:

00	= Alphabetic
01	= Alphanumeric
02	= Numeric Field Data Unsigned Assumed Decimal
03	= Numeric Field Data Signed Assumed Decimal
04	= Numeric Field Data Unsigned Decimal Present
05	= Numeric Field Data Signed Decimal Present
06	= Not Used
07	= Numeric Binary Integer
08	= FORTRAN Single Precision Floating Point Number
09	= FORTRAN Double Precision Floating Point Number

Table 4-2 illustrates examples of all types of data supported by MIRADS and gives input values for the data, the octal representation for the storage of the data in the Data Base, and the output values as they might appear on printed reports.

FIELD DEFINITION CARD 1 (Continued)

DECS IN - Card column 75. Decimals In indicates the number of actual or implied decimal positions (from the right) in the data element described by this card. This value must be 0-8 for DATA TYPES 02-05, otherwise this field should be left blank. Table 4-2 illustrates several examples for Decimals In.

DECS OUT - Card column 76. Decimals Out indicates the number of actual decimal positions (from the right) to be printed by MIRADS for the data element described by this card. This value must be 0-8 for DATA TYPES 02-05 or 01-08 for 08-09; otherwise, this field should be left blank. The value for Decimals Out should never exceed the value for Decimals in DATA TYPES 02-05. Table 4-2 illustrates several examples for Decimals Out.

TLU TABLE NBR - Card columns 77-79 - Indicates the table number in the Table Lookup cards that is to be used for table lookup for the data element described by this card. This value may range from 1 to 999 and should be right-justified, or left blank if the Table Lookup feature is not required by a data element.

G TLU - Card column 80. Global Table Lookup is not presently implemented in MIRADS. The value for this field should be blank.

4.2.5 Table Lookup Card

The Table Lookup card is an optional input card which provides a means for decoding the value of a data element in the Data Base to a more meaningful value for reporting purposes. An example is a coded value of one in a Data Base which is decoded to mean the state ALABAMA for output reports. The Table Lookup card contains both the Data Base coded value and the report value.

Table 4-2. Data Types and Storage

Data Type	Input	Field Width	Deci-mals In (DI)	Storage Octal Code	Deci-mals Out (DO)	Output
00	ABCDEF	6		060710111213		ABCDEF
01	ABC123	6		060710616263		ABC123
02	000666	6	2	606060666666	2	6.66
03	-00000000002	12	0	416060606060 606060606062	0	-2
03	+1234	5	2	4261626364	2	12.34
04	0001.1234567	12	7	606060617561 626364656667	6	1.123456
04	06.366	6	3	606675636666	3	6.366
05	+00000022.22	12	2	426060606060 606262756262	1	22.2
05	-1.001	6	3	416175606061	3	-1.001
07	+4	6		000000000004		4
07	-4	6		777777777773		-4
08	.1E-6	6		151655376247	3	.100-06
08	-.01	6		605270243655	6	-.100000-01
09	.4D-3	12		176564333427 261610313122	8	.40000000-003
09	-.0000001	12		602612240153 124152055724	8	-.10000000-006

Data Type	Description
00	Alphabetic
01	Alphanumeric - COBOL x(n) Format
02	Numeric Field Data Unsigned Assumed Decimal
03	Numeric Field Data Signed Assumed Decimal
04	Numeric Field Data Unsigned Decimal Present
05	Numeric Field Data Signed Decimal Present
06	Not Used
07	Numeric Binary Integer
08	FORTRAN Single Precision Floating Point Number
09	FORTRAN Double Precision Floating Point Number

Each field of the Table Lookup card, as shown in Figure 4-3, is as explained as well as the contents of the fields. One Table Lookup card is required for each coded value for a data element in a Data Base.

TABLE LOOKUP CARD

ACT - Card column 1. Action code indicating one of three possible types of Dictionary input transactions. The action code must be I for Insert, or D for Delete, or M for Modify. The Insert action requires that all fields of the Table Lookup card be completed as described below. The Delete action requires that the ACT, TYPE, TLU TABLE NBR, and TLU DATABASE VALUE fields be completed. The Modify action requires completion of the ACT, TYPE, TLU TABLE NBR, and TLU DATABASE VALUE fields as well as the fields to be altered. Only the values of those fields specified on the Table Lookup card are altered by the Modify action.

TYPE - Card column 2. Card type indicating the type of Dictionary input card. A constant value of T must be used to denote Table Lookup card.

TLU TABLE NBR - Card columns 3-5. Indicates the table number for the data element described by this card and must correspond to the TLU TABLE NBR given on Field Definition Card 1. This value must range from 1 to 999, and should be right justified.

TLU DATABASE VALUE - Card columns 6-14. TLU Data Base value indicates the value contained in the Data Base that corresponds to the TLU Report Value in card columns 15-62 on this card, and should be left justified.

TLU REPORT VALUE - Card columns 15-62. The TLU Report Value corresponds to the TLU Data Base value in card columns 6-14 on this card. This value for a data element will be used on reports in lieu of the coded value when requested on the MIRADS PRINT or FORMAT commands. The allowable characters are A-Z, 0-9, and special characters +, -, $, /, space, and comma. This field should be left justified.

BLANK - Card columns 63-80 should be all blank.

NAME		**TABLE LOOKUP CARD**	SHEET ______ OF ______
ACCOUNT	PHONE		DATE

1	2	3	6	15	63
ACT	TYPE	TLU TABLE NBR	TLU DATA BASE VALUE	TLU – REPORT – VALUE	BLANK
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				

MSFC - Form 422 (Rev. May 1975)

Figure 4-3. Table Lookup Card

SECTION 5 - HOW TO IMPLEMENT A DATA BASE FOR USE WITH MIRADS

5.1 INTRODUCTION

After a program has been written to create a MIRADS Data Base and a Dictionary has been written to define the Data Base, the user is ready to implement a MIRADS application. A User's File Set consisting of five files is required for executing MIRADS inquiries. Section 5 describes the steps required to generate the User's File Set for the first time, and the subsequent steps required to generate a User's File Set for each succeeding time thereafter.

The MIRADS programs used for the implementation cycle are designed to make the process as easy as possible. Dynamic run stream generation and file assignments are provided by the programs so that the user does not have to become acquainted with any files required by the programs except the User's File Set.

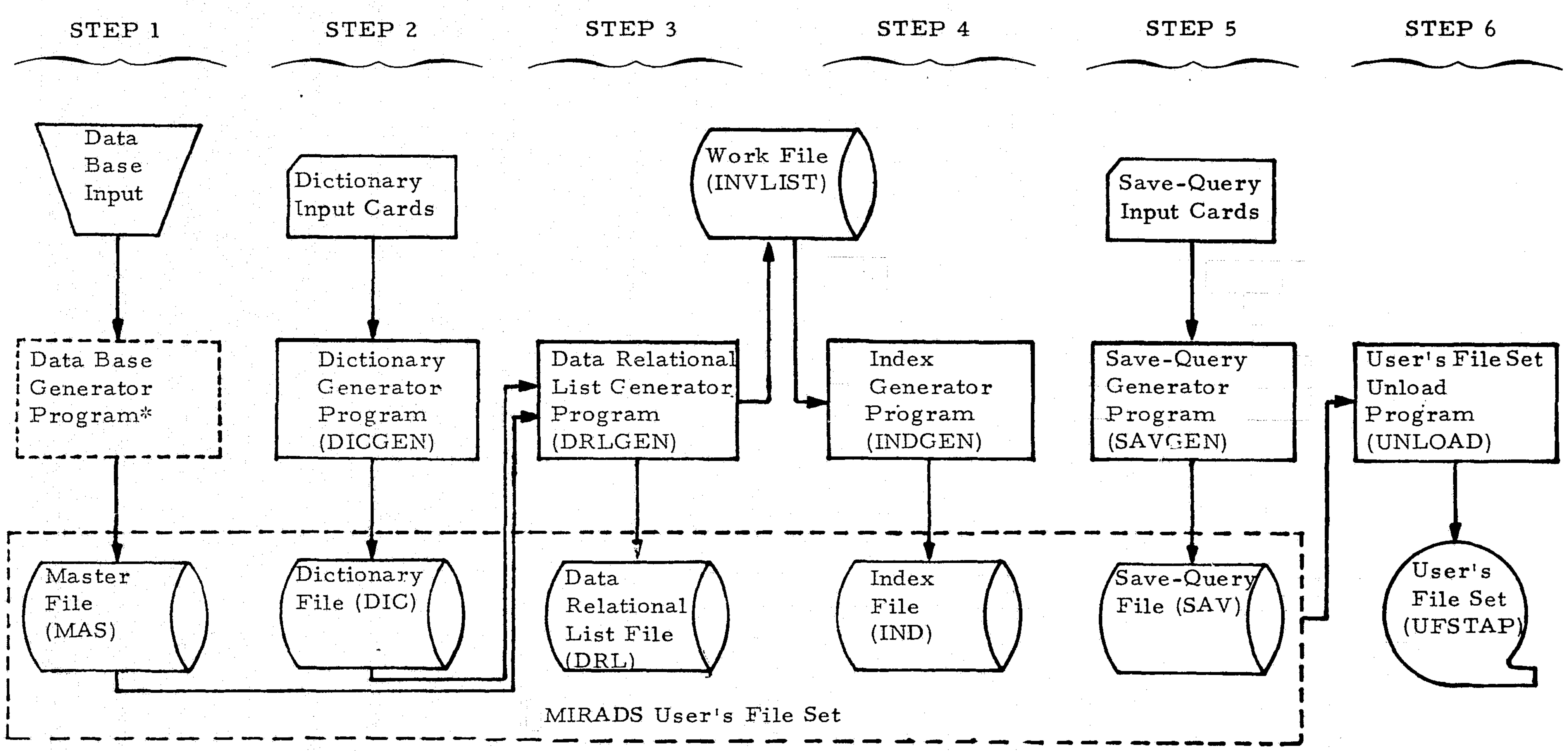
Every possible path in the generation cycle is not discussed because of the many variations available. The initial implementation cycle and one subsequent loading of the User's File Set are discussed in detail. The user who understands these steps should be able to modify the suggested implementation procedures to best suit his needs.

5.2 INITIAL IMPLEMENTATION FOR USER'S FILE SET

The run stream below illustrates the initial procedure for implementing the User's File Set (DIC, SAV, MAS, DRL, and IND files) required for executing MIRADS inquiries. This procedure is used the first time the User's File Set is created. Paragraph 5.3 describes the procedure to follow after initial Data Base implementation. The run stream for implementing the User's File Set is:

```
STEP 1  @RUN
        @ASG, A      MIR*ADS.
        @XQT, C      MIR*ADS. ASGFILS
          ENTER QUALIFIER*FILENAME(CYCLE)
                     (data card naming the Data Base)
        @XQT         USERS. DBGEN
                     (data cards for creating the Data Base)
STEP 2  @XQT, LU     MIR*ADS. DICGEN
          ENTER DICTIONARY CARDS
                     (data cards for creating  the Dictionary)
STEP 3  @XQT         MIR*ADS. DRLGEN
STEP 4  @XQT         MIR*ADS. INDGEN
STEP 5  @XQT         MIR*ADS. SAVGEN
          ENTER QUERY-SET CARDS
                     (data cards for creating Save-Queries)
STEP 6  @XQT,S       MIR*ADS. UNLOAD
          ENTER QUALIFIER*FILENAME(CYCLE)
                     (data card naming the Data Base)
        @FIN
```

A graphic description of the implementation cycle is shown in Figure 5-1.



STEP 1
STEP 2
STEP 3
STEP 4
STEP 5
STEP 6
Data Base Input
Dictionary Input Cards
Work File (INVLIST)
Save-Query Input Cards
Data Base Generator Program*
Dictionary Generator Program (DICGEN)
Data Relational List Generator Program (DRLGEN)
Index Generator Program (INDGEN)
Save-Query Generator Program (SAVGEN)
User's File Set Unload Program (UNLOAD)
Master File (MAS)
Dictionary File (DIC)
Data Relational List File (DRL)
Index File (IND)
Save-Query File (SAV)
User's File Set (UFSTAP)
MIRADS User's File Set

*User written program which creates the Master file using the MIRADS I/O package which is callable from any COBOL, FORTRAN, or Assembly Language program.

Figure 5-1. Initial Data Base Implementation Cycle

1. STEP 1

This run stream assigns the MIRADS Library file, MIR*ADS, to the user's computer run via

```
@ASG,A MIR*ADS.
```

to ensure that it has been loaded onto mass storage and can be accessed. It is assumed that the MIRADS Library file has been cataloged and secured using the UNIVAC 1108 SECURE processor as described in Section 7 of this manual.

Next, the program ASGFILS is executed to assign the necessary files to create the User's File Set.

```
@XQT,C MIR*ADS.ASGFILS
```

The program responds with

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

and the user must enter the filename by which the MIRADS File Set will be known. (The NATIONS file will be used as the example throughout this Section.) After the user enters

```
NATIONS
```

the system will automatically assign the files which are needed to perform MIRADS inquiries. The C option on the @XQT card specifies that cataloged files will be used for the User's File Set. The program equates the files DIC and DICNATIONS, SAV and SAVNATIONS, MAS and MASNATIONS, DRL and DRLNATIONS, and IND and INDNATIONS so that the user's designated File Set, NATIONS, will be used to store the information created by the implementation cycle. The ASGFILS program is documented in Paragraph 6.2.

The creation of the Data Base is the final part of STEP 1 and it is accomplished with a user written program. Even though this is not a MIRADS function, the system requires that the Data Base be written using the MIRADS I/O package and that it be written into the file called MAS. Creation of the MIRADS Data Base is discussed in detail in Section 3 of this manual.

2. STEP 2

At this time, the user is ready to execute the Dictionary generation phase of the implementation cycle. The Dictionary Generator program, DICGEN, generates the Dictionary File, DIC, which describes the contents of the Data Base. The command

```
@XQT, LU MIR*ADS.DICGEN
```

executes the program, and the program responds with the message

```
ENTER DICTIONARY CARDS
```

The Dictionary input cards are entered at this point to generate the MIRADS Dictionary file, DIC. The L option on the @XQT card specifies that Dictionary listings are to be produced in sequence first by field number, then by field name. The U specifies that the passwords are to be printed as part of the Dictionary listings. The MIRADS Dictionary input cards are documented in Section 4, and the DICGEN program is documented in Paragraph 6.3.

3. <u>STEP 3</u>

Following the creation of the Dictionary, the DRL Generator program, DRLGEN, is executed to generate a Data Relations List which describes the hierarchical file structure of the Data Base to MIRADS and stores the results in the DRL file.

```
@XQT MIR*ADS.DRLGEN
```

DRLGEN not only generates the DRL file, but also builds the INVLIST work file which contains the sorted occurrence of each index field in the Data Base. This work file is then used as input to the Index Generator program. Documentation for the DRLGEN program is given in Paragraph 6.4.

4. STEP 4

```
@XQT MIR*ADS.INDGEN
```

The Index Generator program, INDGEN, generates the Index file, IND. This file contains the table of indices which enables rapid response to user inquiries using indexed fields. The INDGEN program is documented in Paragraph 6.5.

5. STEP 5

Saved-elements such as Query sets, print formats, complex compute commands, etc., can be generated at initial Data Base implementation time through the use of the Save Generator program, SAVGEN. The saved elements are stored in the file, SAV, which contains all saved-elements within MIRADS. The command

```
@XQT MIR*ADS.SAVGEN
```

will respond with the message

```
ENTER QUERY-SET CARDS
```

The Saved Query-Set cards are entered at this point to place the saved elements in the User file, SAV. STEP 5 is optional and may be omitted if the user does not wish to enter Saved Query-Sets at this time. The SAVGEN program is documented in Paragraph 6.6.

6. STEP 6

At this point in the RUN, a Data Base has been generated under the file name MAS, a Dictionary under the name DIC, a Data Relation List under the name DRL, an Index file under the name IND, and a Saved Query-Set file under the name SAV. These files should now be placed on a permanent type of storage medium.

The command

```
@XQT,S MIR*ADS.UNLOAD
```

initiates the execution of the UNLOAD program and the program responds with the message

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

The user must enter the same filename as entered for the response to the ASGFILS program in STEP 1. In this case, the response would be

```
NATIONS
```

and the User's File Set will be cataloged on mass storage and Secured to a backup tape that can be loaded using the UNIVAC 1108 SECURE Processor or the MIRADS LOADER program. See Paragraph 6.3 for documentation for the LOADER program. The files of this particular run stream would be cataloged as DICNATIONS, SAVNATIONS, MASNATIONS, DRLNATIONS, and INDNATIONS. The S option in the @XQT card indicates that the User's File Set is to be backed-up using the SECURE Processor. The UNLOAD program is documented in Paragraph 6.7.

The user has now completed the initial implementation cycle for creating the User's File Set, and is ready to use the MIRADS query processing language for querying the Data Base as documented in the MIRADS User's Manual.

5.3 SUBSEQUENT IMPLEMENTATION FOR USER'S FILE SET

Subsequent implementation of a User's File Set as the result of updating the Data Base is not necessarily handled in the same manner as initial file implementation. Steps may be omitted if there have not been any changes made in that particular area. For example:

```
STEP 1          @RUN
                @ASG, A          MIR*ADS.
                @XQT             MIR*ADS. ASGFILS
                  ENTER QUALIFIER*FILENAME(CYCLE)
                NATIONS(+1)
                @XQT             US*ER. UPDATEDB
STEP 2          @ASG, A          DICNATIONS.
                @COPY            DICNATIONS., DIC.
                @FREE            DICNATIONS.
STEP 3          @XQT             MIR*ADS. DRLGEN
STEP 4          @XQT             MIR*ADS. INDGEN
STEP 5          @ASG, A          SAVNATIONS.
                @COPY, P         SAVNATIONS., SAV.
                @FREE            SAVNATIONS.
                @XQT, S          MIR*ADS. UNLOAD
                  ENTER QUALIFIER*FILENAME(CYCLE)
                NATIONS(+1)
                @FIN
```

The run stream above is the most common type of subsequent implementation in which the Data Base changes but the Dictionary remains the same and the Saved Query-Sets are brought forward without any new elements being added.

1. STEP 1

   This run stream assigns the MIRADS Library file, MIR*ADS, to the user's computer run via

   ```
   @ASG, A MIR*ADS.
   ```

   to ensure that it has been loaded onto mass storage and can be accessed. It is assumed that the MIRADS Library file has been cataloged and secured using the UNIVAC 1108 SECURE processor as described in Section 7 of this manual.

Next, the program ASGFILS is executed to assign the necessary files to create the User's File Set.

```
@XQT     MIR*ADS.ASGFILS
```

The program responds with

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

and the user must enter the filename by which the MIRADS File Set will be known. (The NATIONS file will be used as the example throughout this Section.) After the user enters

```
NATIONS(+1)
```

the system will automatically assign the files which are needed to perform MIRADS inquiries. The C option on the @XQT card specifies that cataloged files will be used for the User's File Set. The program equates the files DIC and DICNATIONS(+1), SAV and SAVNATIONS(+1), MAS and MASNATIONS(+1), DRL and DRLNATIONS(+1), and IND and INDNATIONS(+1) so that the user's designated File Set, NATIONS(+1), will be used to store the information created by the implementation cycle. The (+1) cycle is used on the NATIONS Filename so that the existing NATIONS User's File Set, which is cataloged, can be accessed while the updated files are being created. (See Paragraph 2.6.3 of the latest revision of the UNIVAC 1100 series manual UP-4144 for a discussion of File Cycles.) The ASGFILS program is documented in Paragraph 6.2.

The creation of the updated Data Base is the final part of STEP 1 and it is accomplished with a user-written program. Even though this is not a MIRADS function, the system requires that the Data Base be written using the MIRADS I/O package and that it be written into the file called MAS. Creation of the MIRADS Data Base is discussed in detail in Section 3 of this manual.

2. STEP 2

At this time, the user is ready for the Dictionary generation phase of the implementation cycle. In this example, the Dictionary is copied from the SECURE'd file to the new file DIC. The example assumes that the Dictionary does not require any changes. The commands to perform STEP 2 are:

```
@ASG,A      DICNATIONS.
@COPY       DICNATIONS.,DIC.
@FREE       DICNATIONS.
```

The DICNATIONS file must be FREE'd when it is no longer needed, in order to avoid file conflict when the new files are SECURED. Users wishing to update their Dictionary within this cycle should refer to the DICGEN documentation in Paragraph 6.3.

3. STEP 3

Following the creation of the Dictionary, the DRL Generator program, DRLGEN, is executed to generate a Data Relations List which describes the hierarchical file structure of the Data Base to MIRADS and stores the results in the DRL file.

```
@XQT MIR*ADS.DRLGEN
```

DRLGEN not only generates the DRL file, but also builds the INVLIST work file which contains the sorted occurrence of each index field in the Data Base. This work file is then used as input to the Index Generator program. Documentation for the DRLGEN program is given in Paragraph 6.4.

4. STEP 4

```
@XQT MIR*ADS.INDGEN
```

The Index Generator program, INDGEN, generates the Index file, IND. This file contains the table of indices which enables rapid response to user inquiries using indexed fields. The INDGEN program is documented in Paragraph 6.5.

5. STEP 5

The saved elements such as query sets, print formats, complex compute commands, etc., are saved by copying them to the new SAV file. The example assumes that no new elements are to be added to the SAV file during this cycle. The commands to perform STEP 5 are:

```
@ASG, A       SAVNATIONS.
@COPY, P      SAVNATIONS.,SAV.
@FREE         SAVNATIONS.
```

The P option is used on the COPY command to remove deleted elements as the elements are copied. The SAVNATIONS file must be FREE'd when it is no longer needed in order to avoid file conflict when the new files are SECURE'd. Users wishing to add elements to the SAV file within this cycle should refer to the SAVGEN program documentation in Paragraph 6.6.

6. STEP 6

At this point in the RUN a new User's File Set has been generated. A Data Base has been generated under the file name MAS, a Dictionary under the name DIC, a Data Relational List under the name DRL, an Index file under the name IND, and a Save-Query-Set file under the name SAV. These files should now be placed on a permanent type of storage medium.

The command

```
@XQT, S MIR*ADS. UNLOAD
```

initiates the execution of the UNLOAD program and the program responds with the message

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

The user must enter the same filename as entered for the response to the ASGFILS program in STEP 1. In this case, the response would be

```
NATIONS(+1)
```

and the User's File Set will be cataloged on mass storage and secured to a backup tape that can be loaded using the UNIVAC 1108 SECURE Processor or the MIRADS LOADER program. See Paragraph 6.3 for documentation for the LOADER program. The files of this particular run stream would be cataloged as DICNATIONS, SAVNATIONS, MASNATIONS, DRLNATIONS, and INDNATIONS. The S option in the @XQT card indicates that the User's File Set is to be backed-up using the SECURE

Processor. The UNLOAD program is documented in Paragraph 6.7.

The user has now completed the implementation cycle for creating an updated User's File Set. Users that query the NATIONS Data Base after execution of the UNLOAD program will access the updated User's File Set although the old File Set is still cataloged. The old User's File Set should be deleted with the following commands after the new User's File Set has been checked out:

```
@DELETE, C     DICNATIONS(-1).
@DELETE, C     SAVNATIONS(-1).
@DELETE, C     MASNATIONS(-1).
@DELETE, C     DRLNATIONS(-1).
@DELETE, C     INDNATIONS(-1).
```

SECTION 6 - MIRADS IMPLEMENTATION PROGRAMS

6.1 INTRODUCTION

The MIRADS implementation programs are used to create a User's File Set which is necessary for the implementation of a MIRADS Data Base. The functions performed by the implementation programs include assigning all files necessary to make a computer run to load a Data Base; creation of the Dictionary, Data Relational List, Index, and Save-Query files; and unloading the newly created User's File Set to tape for backup purposes. The following paragraphs describe in detail the individual procedures for the execution of the implementation programs.

6.2 ASGFILS FILE ASSIGNMENT PROGRAM

The user must make a decision at the beginning of the implementation cycle whether to use cataloged or temporary files during the creation of the User's File Set. Cataloged files are used for the File Set when the Data Base is used frequently enough to reside permanently on mass storage. Temporary files, on the other hand, are used for those applications that will reside permanently on tape and will be copied to mass storage only when they are needed. The program ASGFILS is used to assign the files of the User's File Set and to establish a relationship between the short internal filenames which are used for the physical storage of the User's File Set (for example, @USE IND, INDNATIONS). The command to initiate execution of the program is:

```
@XQT MIR*ADS.ASGFILS
```

and the program responds with the message

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

1. QUALIFIER

   This is an optional 1-to-12-character entry that is an extension to the basic name of the file. If the entry is omitted, the implied qualifier will be used according to UNIVAC 1108 Executive Operating System default conventions.

2. FILENAME

   The FILENAME is a required one-to-nine-character basic Filename by which the User's File Set will be referenced.

3. CYCLE

   This is an optional entry used to differentiate successive updates of the file. It is normally used after the initial implementation cycle to permit users to query an existing User's File Set while implementing an updated File Set.

The valid forms of entries are:

```
STANDARD
QUAL*FILE(+1)
QUAL*FILE
     *FILE(+1)
     *FILE
      FILE(+1)
      FILE
```

The STANDARD entry indicates that temporary files are to be used in the implementation cycle. The MIRADS standard User's File Set of temporary files will be assigned by the following commands to the EXECUTIVE:

```
@ASG, T      DIC. , F2//POS/5
@ASG, T      SAV. , F2//POS/5
@ASG, T      MAS. , F2//POS/500
@ASG, T      DRL. , F2//POS/50
@ASG, T      IND. , F2//POS/50
```

The other entries assign the User's File Set as cataloged files so that the implementation programs will write directly into these files. For purposes of illustration, the qualifier is QUAL and the filename is FILE in all entries. The (+1) cycle entry is used to assure the integrity of any files currently cataloged with the same qualifier and filenames, and to permit other users to access those files while the new User's File Set is being generated. The entry

```
QUAL*FILE(+1)
```

will result in the following commands to the EXEC:

```
@ASG,UPV      QUAL*DICFILE(+1).,F2//POS/5
@ASG,UPV      QUAL*SAVFILE(+1).,F2//POS/5
@ASG,UPV      QUAL*MASFILE(+1).,F2//POS/500
@ASG,UPV      QUAL*DRLFILE(+1).,F2//POS/50
@ASG,UPV      QUAL*INDFILE(+1).,F2//POS/50
@USE          DIC.,QUAL*DICFILE(+1).
@USE          SAV.,QUAL*SAVFILE(+1).
@USE          MAS.,QUAL*MASFILE(+1).
@USE          DRL.,QUAL*DRLFILE(+1).
@USE          IND.,QUAL*INDFILE(+1).
```

6.3 DICGEN DICTIONARY GENERATION PROGRAM

6.3.1 Creating the Dictionary

The Dictionary Generator is an edit/update program that takes the Dictionary input cards and creates a file, DIC, which describes the Data Base to MIRADS. The Dictionary cards are edited for verification of format and content upon input, and these cards may be used to generate a new Dictionary or modify an existing Dictionary. The command to execute the Dictionary Generator is:

```
@XQT,Options    MIR*ADS.DICGEN
```

Options	Function
N	Inhibit the Dictionary listing but print all diagnostic error messages.
S	Produce a Dictionary listing in sequence by field number.

Options	Function
L	Produce Dictionary listings in sequence by field number and by field name.
U	Print a listing of all user password entries. This option is to be used in conjunction with the L or S options.

If there are no options specified on the @XQT command, the program will respond with the message

```
ENTER NONE, SHORT OR LONG
```

to solicit the type of Dictionary listing desired. NONE will inhibit the Dictionary listing but will print all diagnostic messages (N option), SHORT will produce a Dictionary listing in field number sequence (S option), and LONG will produce Dictionary listings in sequence by field number and by field name (L option).

When DICGEN establishes the type of listing the user desires, the program responds with the message

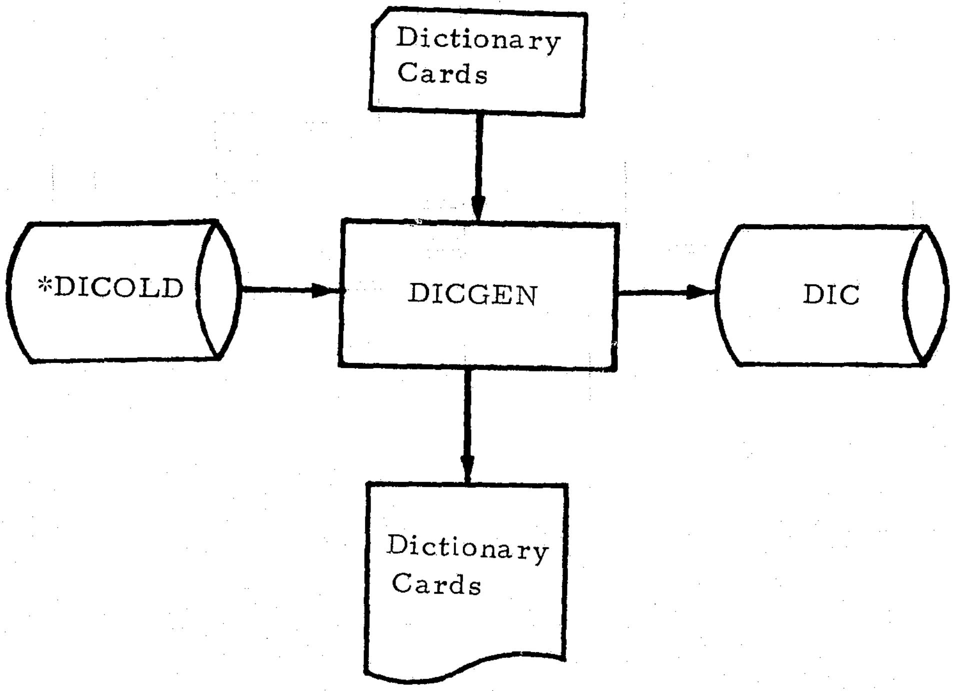
```
ENTER DICTIONARY CARDS
```

The user must enter the Dictionary cards at this time. The Dictionary input cards are described in Section 4.

6.3.2 Updating the Dictionary

DICGEN is an edit/update program and provides for modifying a Dictionary once it has been created. New records can be inserted into an existing Dictionary and/or current records can be modified to produce an updated Dictionary. To prepare for a Dictionary update, the current Dictionary must be placed in a file called DICOLD. Execution of DICGEN will read the Dictionary in DICOLD, update it with insert or modify cards, and write the updated Dictionary to the file, DIC. See Figure 6-1. Using the NATIONS File Set as a sample application, the following portion of a run stream illustrates the commands as they might be used in an implementation cycle:

```
@RUN
@ASG,A          MIR*ADS.
@ASG,A          DICNATIONS.
@USE            DICOLD,DICNATIONS
@XQT            MIR*ADS.ASGF[illegible]S
NATIONS(+1)
.
.
@XQT,SU         MIR*ADS.DICGEN
.
.
@FREE           DICNATIONS.
@XQT            MIR*ADS.DRLGEN
.
.
.
```



*The DICOLD file is only used when an existing Dictionary file is to be updated.

Figure 6-1. Dictionary Generation

This run stream assumes that the NATIONS File Set is cataloged and that the new File Set will be loaded into the +1 or next highest cycle. The commands

```
@ASG, A      DICNATIONS.
@USE         DICOLD, DICNATIONS
```

ensure that the Dictionary will be loaded and that it will be accessible by the DICGEN program when the DICOLD file is referenced. The command

```
@FREE      DICNATIONS.
```

is placed after execution of DICGEN so there will be no conflict with the +1 cycle when the new User's File Set is to be saved.

6.3.3 Frequently asked Questions about Generating a Dictionary

1. Can a user make changes to an existing Dictionary without repeating the entire implementation cycle for a Data Base?

   ANSWER: Yes and No. Corrections can be made to the Table Lookup cards and the Field Definition cards provided the data element being described is not an indexed field or does not redefine an indexed field in any manner. After the corrections have been made to the Dictionary cards, the Dictionary generation program, DICGEN, should be executed. The new Dictionary file, DIC, created by this program should then be used to replace the old Dictionary file.

2. Can the Dictionary generation program be run as a stand-alone program to verify the Dictionary input cards when they are initially created?

   ANSWER: The DICGEN program can be run as a stand-alone program to edit and verify the Dictionary input cards for correctness. When running the program in this manner, the execution of the ASGFILS program is not necessary since the DICGEN program will dynamically assign a temporary output file named DIC for creating or storing the new Dictionary file.

6.4 DRLGEN DATA RELATIONAL LIST GENERATION PROGRAM

6.4.1 Introduction

The primary function of the Data Relational List (DRL) program is to create the DRL file of the User's File Set. The DRL file describes the hierarchical structure of the Data Base to the MIRADS processing programs. The hierarchical structure is automatically created by the DRLGEN program using information extracted from the Dictionary and Data Base files. The DRL file enables MIRADS to query the Data Base at any level in the hierarchical structure and be capable of operating on the selected record's owner and member records.

The secondary function of DRLGEN is to select and sort all occurrences of each indexed field in the Data Base, and build the INVLIST work file. This file is the only input to the Index Generator, INDGEN, and execution of DRLGEN should always be followed by execution of INDGEN. The command to execute DRLGEN is:

```
@XQT      MIR*ADS.DRLGEN
```

6.4.2 Time Estimating and Efficiency

DRLGEN uses approximately three-fourths of the processing time that it takes to implement a Data Base. Eighty percent of this time is used in sorting the occurrences of each data value for indexed fields, so the number of indexed fields and the number of physical records in the Data Base determine the amount of CPU-time used for implementing a Data Base. The number of records sorted by the DRLGEN program is equal to the number of indexed fields multiplied by the number of physical records in the Data Base. For estimating total CPU-time required to implement a Data Base, use 1-minute CPU-time for each 25,000 sort records.

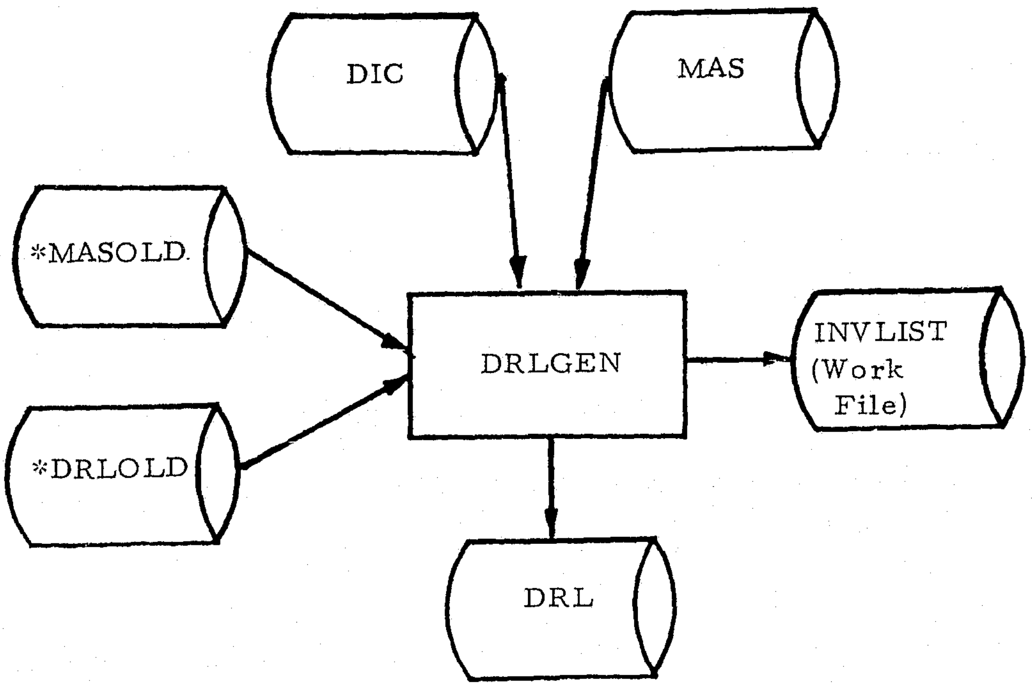
The efficiency of the sort within DRLGEN will affect the CPU-time slightly and will greatly affect the elapsed time for DRLGEN program execution. File assignments are made dynamically within the DRLGEN program in order to keep the run stream simple and easy to implement. The File Name Card of the Dictionary contains a field stating the total number of Data Base Records. This value is used for determining dynamic sort file assignments within DRLGEN and must properly reflect the maximum size of the Data Base in order to maintain program efficiency.

6.4.3 Reordering the DRL File and Data Base

Users that utilize the MIRADS Update command capability may eventually want to reorder the Data Base and eliminate overflow records and pointers in the DRL file that affect query response time in MIRADS. Both the DRL and Data Base (MAS) files can be reordered by the

DRLGEN program. The existing DRL file must be copied into a file called DRLOLD and the Data Base must be copied into a file called MASOLD for DRLGEN to initiate the reordering process. The program will then read the DRLOLD file and write a new DRL file in a reordered sequence. The DRLOLD records are used to access the MASOLD records and write the new MAS file in the reordered sequence. See Figure 6-2. Since the relative addresses of records in these files are changed, the execution of the INDGEN program, using the INVLIST work file as input, must follow to generate a new IND file. Using the NATIONS File Set as a sample application, the following portion of a run stream illustrates the commands that might be used in an implementation cycle:

```
@RUN
@ASG,A          MIR*ADS.
@ASG,A          DICNATIONS.
@ASG,A          SAVNATIONS.
@ASG,A          MASNATIONS.
@ASG,A          DRLNATIONS.
@USE            MASOLD,MASNATIONS
@USE            DRLOLD,DRLNATIONS
@XQT            MIR*ADS.ASGFILS
NATIONS(+1)
@COPY           DICNATIONS.,DIC.
@COPY,P         SAVNATIONS.,SAV.
@XQT            MIR*ADS.DRLGEN
@FREE           MASNATIONS.
@FREE           DRLNATIONS.
@FREE           DICNATIONS.
@FREE           SAVNATIONS.
@XQT            MIR*ADS.INDGEN
@XQT,S          MIR*ADS.UNLOAD
NATIONS(+1)
@FIN
```



*The MASOLD and DRLOLD files are only used when the DRL and Data Base files are being reordered.

Figure 6-2. DRL Generation

This run stream assumes that the NATIONS File Set is cataloged and that the new File Set will be loaded into the +1 or next highest cycle. The commands

```
@ASG, A     DICNATIONS.
@ASG, A     SAVNATIONS.
@ASG, A     MASNATIONS.
@ASG, A     DRLNATIONS.
```

ensure that the current Dictionary (DIC), Save-Query (SAV), Data Base (MAS), and DRL files are loaded and attached to the run. The commands

```
@USE      MASOLD, MASNATIONS
@USE      DRLOLD, DRLNATIONS
```

cause the DRLGEN program to access the existing Data Base and DRL files when MASOLD and DRLOLD are referenced. The commands

```
@COPY       DICNATIONS., DIC.
@COPY, P    SAVNATIONS., SAV.
```

are placed after execution of ASGFILS in order to copy the existing Dicationary and Save-Query files into the +1 cycle of the User's File Set so they will be referenced properly by the DRLGEN and UNLOAD programs. The commands

```
@FREE    DICNATIONS.
@FREE    SAVNATIONS.
@FREE    MASNATIONS.
@FREE    DRLNATIONS.
```

are placed after execution of DRLGEN so that there will be no conflict with the +1 cycle when the new User File Set is to be saved.

To estimate times for reordering the Data Base and DRL file, use the guidelines of Paragraph 6.4.2.

6.5 INDGEN INDEX GENERATION PROGRAM

The function of the Index Generator program is to create the Index file, IND, which provides MIRADS with rapid access to Data Base records containing indexed fields. The only input to INDGEN is the INVLIST work file which is created by the DRLGEN program. The INVLIST file contains the sorted values of each Data Base record for each indexed field described in the Dictionary. The INDGEN program builds the IND file with an Indexed Sequential File organization. See Figure 6-3. The IND file then provides direct access to Data Base records through the Data Relational List, DRL.

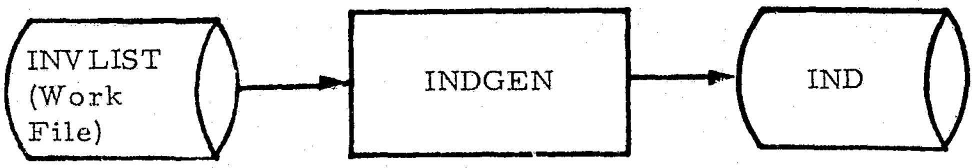


Figure 6-3. Index Generation

The command to execute the Index Generator is

```
@XQT      MIR*ADS.INDGEN
```

and it must always be executed following the DRL Generator program even if there are no indexed fields in a Data Base.

Proper indexing of Data Base fields is the key to rapid response and efficient processing of MIRADS inquiries. Selection of the wrong fields for indexing, or selection of too few indexed fields can result in excessive sequential searching of the Data Base. Similarly, selection of too many indexed fields can result in high implementation costs in the DRLGEN program and generate a larger Index file, IND, which would require additional search time. The type of inquiries that will be made on the Data Base will best determine the fields that should be indexed.

6.6 SAVGEN SAVE ELEMENT GENERATION PROGRAM

6.6.1 Creating the Save Elements

The Save-Element Generator program provides a means for the user to enter query sets or complex MIRADS commands into the Save-Element file, SAV, prior to creating a permanent backup of the User's File Set. The SAVGEN program is provided as a convenience for the user and places the entries into the SAV file as they are received. The commands are not edited or checked for form or content; therefore, there is no assurance that the query sets will execute as entered into the SAV file. The command to execute the Save Element Generator is

```
@XQT      MIR*ADS.SAVGEN
```

and the program responds with the message

```
ENTER QUERY-SET CARDS
```

The first entry to the program identifies that this is the start of a Query-Set and provides a name for the query set to be entered. The following entries identify a complete query set for the NATIONS File Set.

SAV. CITIES	Indicates the beginning of a Query-Set named CITIES
QUERY, CITY PRESENT. SORT, STATE, CITY. PRINT, STATE GROUP 1, CITY.	Entries for the Query-Set
@END	Indicates the end of the Query-Set named CITIES

The entry SAV. identifies a new query set being entered into the SAV file. Immediately following SAV. is a 1-to-12-character entry used to name the Query-Set. The MIRADS QUERY, SORT, and PRINT commands will be placed in the SAV file as they are entered and will be known as the CITIES Query-Set. The end of one query set entry and the beginning of another is signified by another SAV. entry or by an EXEC VIII Control card such as @END.

6.6.2 Updating the SAVE Element File

The SAVGEN program provides a means for the user to retain prior Save Elements with each subsequent loading of new User's File Sets. The SAV file with Save Elements to be retained must be in the file SAVOLD prior to execution of SAVGEN. The Save-Elements in SAVOLD will be copied to the new SAV file, and new Query-Sets can also be added through the normal entry procedure. Using the NATIONS File Set as a sample application, the following portion of a run stream illustrates the commands that might be used in an implementation cycle:

```
@RUN
@ASG, A        MIR*ADS.
@ASG, A        SAVNATIONS.
@USE           SAVOLD, SAVNATIONS
@XQT           MIRADS*ASGFILS
NATIONS(+1)
.
.
.
@XQT           MIR*ADS.SAVGEN
SAV.STATES
Q,STATE P.
S, P-RANK D.
P, STATE, S-POP, P-RANK.
@FREE          SAVNATIONS.
.
.
.
```

This run stream assumes that the NATIONS File Set is cataloged and that the new User's File Set will be loaded into the +1 or next highest cycle. The commands

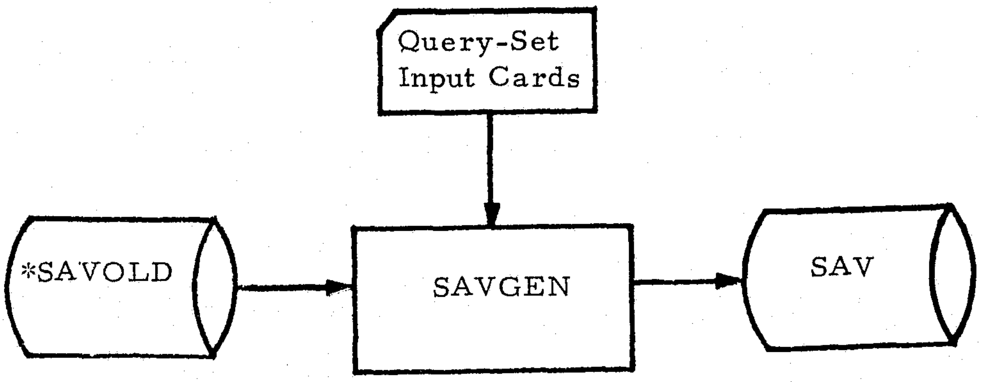
```
@ASG, A     SAVNATIONS
@USE        SAVOLD, SAVNATIONS
```

ensure that the Save-Element File will be loaded and that it will be accessed when SAVOLD is referenced. Upon execution of SAVGEN, the saved elements of SAVNATIONS (SAVOLD) will be copied to the new SAV file and a new Query-Set, STATES, will be inserted into the new SAV file. The command

```
@FREE     SAVNATIONS.
```

c-2

follows the execution of SAVGEN so there will be no conflict with the +1 cycle when the new User's File Set is to be saved. See Figure 6-4.



*The SAVOLD file is only required when an existing Save-Element file is to be updated.

Figure 6-4. SAV File Generation

6.7 UNLOAD PROGRAM

6.7.1 Introduction

The UNLOAD program is a utility program that generates a run stream to create backup copies of the User's File Set. The command to execute the UNLOAD program is

```
@XQT,Options    MIR*ADS.UNLOAD
```

Options	Function
S	Use the UNIVAC 1108 SECURE processor to create a magnetic tape backup for the User's File Set and to log the User's File Set into the EXEC VIII Master File Directory. This option cannot be used when

Options	Function
	temporary files have been used for the Data Base implementation cycle.
R	Rollout the User's File Set to a magnetic tape in UNIVAC 1108 @COPY, G format.

If there are no options specified in the @XQT card, the program will default to the S option. If both options are specified, the program will rollout the User's File Set to magnetic tape and then create a SECURE'd backup User's File Set. As soon as UNLOAD is executed, it responds with the message

ENTER QUALIFIER*FILENAME(CYCLE)

1. QUALIFIER

   This is an optional 1-to-12-character entry that is an extension to the basic name of the file. If the entry is omitted, the implied qualifier will be used according to UNIVAC 1108 Executive Operationg System default conventions.

2. FILENAME

   FILENAME is a required one-to-nine-character basic Filename by which the User's File Set is referenced.

3. CYCLE

   This is an optional entry to differentiate successive updates of the file. It is normally used after the initial implementation cycle to permit users to query an existing User's File Set while implementing an Updated File Set.

The valid forms of entries are:

```
STANDARD
QUAL*FILE(+1)
QUAL*FILE
     *FILE(+1)
     *FILE
      FILE(+1)
      FILE
```

6.7.2 SECUREing the User's File Set with UNLOAD

If the S option is specified on the UNLOAD execute card, the procedure for using the SECURE processor is initiated.

After the Filename entry is made, the program will FREE the files of the User's File Set in order to place the cataloged files into the UNIVAC 1108 Master File Directory. The User's File Set is then assigned with exclusive use of the files to prevent other users from attaching the files while they are being SECURE'd. The Secure Backup tape is then assigned with the following command:

```
@ASG, NT OBACKUP, 8C, SAVE04 . MIRADS USERS FILE SET
```

followed by the SECURE processor directives. After the files have been SECURE'd, the exclusive use attachment is removed from the User's File Set but the files are still assigned to the run. The OBACKUP tape is released as the final step of the SECURE process. A user entry of

```
QUAL*FILE(+1)
```

will result in the execution of the following run stream:

```
@FREE           QUAL*DICFILE(+1).
@FREE           QUAL*SAVFILE(+1).
@FREE           QUAL*MASFILE(+1).
@FREE           QUAL*DRLFILE(+1).
@FREE           QUAL*INDFILE(+1).
@ASG, AX        QUAL*DICFILE.
@ASG, AX        QUAL*SAVFILE.
@ASG, AX        QUAL*MASFILE.
@ASG, AX        QUAL*DRLFILE.
@ASG, AX        QUAL*INDFILE.
@ASG, NT        OBACKUP, 8C, SAVE04 . MIRADS USERS FILE SET
@SECURE, ILC
SAVE ALL FILES;
          QUAL*DICFILEb;
          QUAL*SAVFILEb;
          QUAL*MASFILEb;
          QUAL*DRLFILEb;
          QUAL*INDFILEb;
 TO OBACKUP
 @FREE, X       QUAL*DICFILE.
 @FREE, X       QUAL*SAVFILE.
 @FREE, X       QUAL*MASFILE.
 @FREE, X       QUAL*DRLFILE.
 @FREE, X       QUAL*INDFILE.
 @FREE          OBACKUP.
```

If no errors are encountered, the User's File Set is copied to this tape in SECURE format. Upon completion, SECURE generates an output summary listing under the following format:

```
SECURE SUMMARY LISTING:
QUAL*DICFILE      Saved at 09:15:31 to File 1 Reel 25660
QUAL*SAVFILE      Saved at 09:17:14 to File 3 Reel 25660
QUAL*MASFILE      Saved at 09:17:35 to File 4 Reel 25660
QUAL*DRLFILE      Saved at 09:17:11 to File 2 Reel 25660
QUAL*INDFILE      Saved at 09:17:47 to File 5 Reel 25660
END OF SECURE - TIME 1.228 SECONDS
```

6.7.3 Rollout of the User's File Set Using UNLOAD

The R option in the UNLOAD execute command generates a run stream that will copy the User's File Set to magnetic tape in UNIVAC 1108 @COPY,G format. This dynamically assigns a backup tape with the following command

```
@ASG,T UFSTAP,8C,SAVE04 . MIRADS USERS FILE SET
```

If temporary files are being used and STANDARD is the Filename entry, UNLOAD generates the following run stream to rollout the User's File Set to the tape.

```
@COPY,GM     DIC.,UFSTAP.
@COPY,GM     SAV.,UFSTAP.
@COPY,GM     MAS.,UFSTAP.
@COPY,GM     DRL.,UFSTAP.
@COPY,GM     IND.,UFSTAP.
```

If cataloged files are being used, the Filename entry is used to rollout the User's File Set to tape, then the files are FREE'd so they will be immediately available to other users. The user entry of

```
QUAL*FILE
```

will result in the following run stream:

```
@ASG, T  UFSTAP, 8C, SAVE04 .  MIRADS USERS FILE SET
@COPY, GM      QUAL*DICFILE., UFSTAP.
@COPY, GM      QUAL*SAVFILE., UFSTAP.
@COPY, GM      QUAL*MASFILE., UFSTAP.
@COPY, GM      QUAL*DRLFILE., UFSTAP.
@COPY, GM      QUAL*INDFILE., UFSTAP.
@FREE          QUAL*DICFILE.
@FREE          QUAL*SAVFILE.
@FREE          QUAL*MASFILE.
@FREE          QUAL*DRLFILE.
@FREE          QUAL*INDFILE.
```

The R option of UNLOAD also generates the following commands to provide the user with the reel number of the backup tape

```
@XQT  MIR*ADS.TPNO
UFSTAP
```

and the user will receive a response of the following format

```
UFSTAP = 25661
```

The tape UFSTAP is neither rewound nor FREE'd at the completion of UNLOAD execution in order for it to be available to the user for providing additional backup files.

6.7.4 Frequently asked Questions about Unloading a Data Base

1. Is UNLOAD a required part of the implementation cycle?

   ANSWER: No. It is provided as a convenience to the user. You may provide backup capability in any manner you desire.

2. The assign format for tapes UFSTAP and OBACKUP do not conform to my computer installation conventions for tape retention. Do I have to modify UNLOAD in order to be able to use it?

   ANSWER: The user can override any files that are dynamically assigned within any program of the MIRADS System. All he has to do is to assign the file prior to executing the program. For example, the commands

   ```
   @ASG,T      MYFILE,8C,11407
   @USE        UFSTAP,MYFILE
   @XQT,R      MIR*ADS.UNLOAD
   ```

   couid be used to override the UNLOAD assignment of the UFSTAP tape and permit the user to provide his own file name for the tape.

3. Can the UNLOAD program be used external to the implementation cycle?

   ANSWER: Yes. As a utility program, UNLOAD can be used at any time to rollout a User's File Set to tape, or to SECURE a cataloged User's File Set to tape.

6.8 LOADER PROGRAM

6.8.1 Introduction

The LOADER program is a utility program that generates a run stream to load the MIRADS User's File Set from magnetic tape to a mass storage device. The program loads files from either the UNIVAC 1108 Rollout or Secure formats. By executing this program as the first task in a run stream, the user can ensure that the User's File Set is loaded and available for use by the MIRADS System. The LOADER program performs all functions necessary to catalog, load, and assign the User's File Set to the user's run. The command for executing the LOADER program is

```
@XQT,Options   MIR*ADS.LOADER
```

Options	Function
S	Use the UNIVAC 1108 SECURE processor to load the User's File Set to mass storage from a magnetic tape and log the User's File Set into the EXEC VIII Master File Directory.
R	Use the UNIVAC 1108 FURPUR @COPY,G format to rollin or load the User's File Set to mass storage from a magnetic tape and log the User's File Set into the EXEC VIII Master File Directory.

If there are no options, or more than one option specified in the @XQT card, the program will default to the S option.

6.8.2 Loading the SECURE'd User's File Set

After the LOADER program has been executed with the S option, it responds with the message

```
ENTER QUALIFIER*FILENAME
```

1. QUALIFIER

   This is a required 1-to-12-character entry that is an extension to the basic name of the file.

2. FILENAME

   This is a required one-to-nine-character basic Filename by which the User's File Set is referenced.

The only valid form of entry is

QUAL*FILE

After the QUALIFIER*FILENAME has been entered, the program then determines if the files are already loaded. If they are, then LOADER program terminates execution normally and no action is taken. If the files are not loaded, the program determines if the reel number(s) of the magnetic tape containing the SECURE'd files is present in the EXEC VIII Master File Directory. If it is not present, the program responds with

ENTER SECURE TAPE REEL NUMBER(S)

The user must enter a one-to-five-digit reel number for each tape with multiple reel numbers being separated by slashes (/). After the reel number(s) has been determined, the LOADER program loads the User's File Set and the loading process for SECURE'd files is complete.

The following example illustrates the use of the LOADER program for a SECURE'd User's File Set, and the run stream generated by the program to load the files

```
@XQT, S        MIR*ADS. LOADER
    ENTER QUALIFIER*FILENAME
QUAL*FILE
    ENTER SECURE TAPE REEL NUMBER(S)
12345/67890
@ASG, NT       IBACKUP, 8C, 12345/67890
@SECURE, ILC
 IBACKUP = 12345/67890                        Run stream generated
LOAD PROJECT QUAL;                            by LOADER from
FROM  IBACKUP                                 above input.
@END
@FREE        IBACKUP
```

The IBACKUP tape(s) is FREE'd as the final step in the load process.

6. 8. 3 Rolling-In the User's File Set

After the LOADER program has been executed with the R option, it responds with the message

ENTER QUALIFIER*FILENAME(CYCLE)

1. QUALIFIER

   This is an optional 1-to-12-character entry that is an extension to the basic name of the file. If the entry is omitted, the implied qualifier will be used according to UNIVAC 1108 Executive Operating System default conventions.

2. FILENAME

This is a required one-to-nine-character basic Filename by which the User's File Set is referenced.

3. CYCLE

CYCLE is an optional entry to be used to differentiate succesive updates of the file. Cycle number will normally be omitted when using the LOADER program.

The valid forms of entries are:

```
STANDARD
QUAL*FILE(+1)
QUAL*FILE
     *FILE(+1)
     *FILE
     *FILE(+1)
      FILE
```

The user can load the User's File Set into temporary files by responding with the word STANDARD; otherwise, the File Set will be loaded into cataloged files. After the QUALIFIER*FILENAME(CYCLE) has been entered, the program then determines if the files are loaded. If they are, no action is taken and the program terminates execution normally. If the files are not loaded the program responds

```
ENTER ROLLIN TAPE REEL NUMBER(S)
```

The user must enter a one-to-five-digit reel number for each tape with multiple reel numbers being separated by slashes (/). After the reel number(s) has been determined, the LOADER program loads the User's File Set and the loading process is complete.

The following examples illustrate the use of the LOADER program for rolling-in the User's File Set, and the run streams generated by the program to load the files:

EXAMPLE 1 FOR TEMPORARY FILES:

```
@XQT, R      MIR*ADS. LOADER
    ENTER QUALIFIER*FILENAME(CYCLE)
STANDARD
    ENTER ROLLIN TAPE REEL NUMBER(S)
12345
@ASG, T      DIC., F2//POS/5  .
@ASG, T      SAV., F2//POS/5  .
@ASG, T      MAS., F2//POS/500  .
@AST, T      DRL., F2//POS/50  .
@AST, T      IND., F2//POS/50  .          Run stream generated
@ASG, T  UFSTAP, 8C,  12345.              by LOADER from above
@COPY, G     UFSTAP., DIC.  .             input.
@COPY, G     UFSTAP., SAV.  .
@COPY, G     UFSTAP., MAS.  .
@COPY, G     UFSTAP., DRL.  .
@COPY, G     UFSTAP., IND.  .
@FREE  UFSTAP. .
```

EXAMPLE 2 FOR CATALOGED FILES:

```
@XQT,R      MIR*ADS.LOADER
    ENTER QUALIFIER*FILENAME(CYCLE)
QUAL*FILE
    ENTER ROLLIN TAPE REEL NUMBER(S)
12345/67890
@ASG,UPV    QUAL*DICFILE.,F2//POS/5    .
@ASG,UPV    QUAL*SAVFILE.,F2//POS/5    .
@ASG,UPV    QUAL*MASFILE.,F2//POS/500 .
@ASG,UPV    QUAL*DRLFILE.,F2//POS/50   .
@ASG,UPV    QUAL*INDFILE.,F2//POS/50   .
@ASG,T  UFSTAP,8C,12345/67890   .
@COPY,G     UFSTAP.,QUAL*DICFILE.    .        Run stream
@COPY,G     UFSTAP.,QUAL*SAVFILE.    .        generated by
@COPY,G     UFSTAP.,QUAL*MASFILE.    .        LOADER from
@COPY,G     UFSTAP.,QUAL*DRLFILE.    .        above input.
@COPY,G     UFSTAP.,QUAL*INDFILE.    .
@FREE       QUAL*DICFILE.  .
@FREE       QUAL*SAVFILE.  .
@FREE       QUAL*MASFILE.  .
@FREE       QUAL*DRLFILE.  .
@FREE       QUAL*INDFILE.  .
@FREE  UFSTAP.  .
```

In both examples, the User's File Set Tape (UFSTAP) is FREE'd as the final step in the load process.

SECTION 7 - MIRADS UTILITY PROGRAMS/SUBROUTINES

7.1 INTRODUCTION

A set of utility programs/subroutines has been developed to enable users to perform various file manipulation functions that may be required in conjunction with the use of MIRADS. These functions include reading and writing of records on mass storage, reading card reader files, formatted dumps of mass storage files, etc. The utilities are contained in the MIRADS Library file named MIR*ADS.

The following paragraphs explain the functions of each program/subroutine and illustrate the procedures required to use them.

7.2 IOPKG INPUT/OUTPUT SUBROUTINE

The MIRADS IOPKG subroutine is a UNIVAC 1108 mass storage-oriented I/O package designed to provide efficient processing of data in either random or sequential order. The package allows the user to select single or double I/O buffers so that emphasis may be placed on either program size or speed. The double buffers require more core memory but allow I/O operations to overlap with internal processing. Random processing is supported only in the single buffer mode. Both modes support blocked and unblocked files. The package may be accessed from UNIVAC 1108 COBOL or FORTRAN as follows:

From	Compiler Verb
DOD COBOL	ENTER
FD COBOL	ENTER
ASCII COBOL	CALL
FORTRAN V	CALL

IOPKG contains several entry points, each of which performs a specific function. These functions and the rules for their usage will be explained in the following paragraphs.

7.2.1 OPENS Entry Point

This entry point opens a mass storage file for subsequent processing by establishing a 15-word File-Control-Table (FCT).

DOD AND FD COBOL

```
ENTER FORTRAN OPENS SUBROUTINE REFERENCING
FILENAME RECSIZE BLKSIZE BUFFER NUMBUFS FILUSE.
```

ASCII COBOL

```
CALL 'OPENS' USING FILE RECSIZE BLKSIZE BUFFER NUMBUFS
FILUSE.
```

FORTRAN V

```
CALL OPENS (FILENAME RECSIZE BLKSIZE BUFFER NUMBUFS
FILUSE).
```

1. FILENAME

   Two words containing a one-to-twelve fieldata character(s) filename, left-justified, and space-filled to the right.

2. RECSIZE

One word containing a binary number representing the record size in words.

3. BLKSIZE

One word containing a binary number representing the number of records per block.

4. BUFFER

A read/write file buffer area in the users program for exclusive use by the IOPKG subroutine. Buffer size in words is calculated as follows:

Unblocked file with single buffer
  15 words

Unblocked file with double buffers
  (Recsize x 2) + 15

Blocked file with single buffer
  (Recsize x Blksize) + 15

Blocked file with double buffers
  ((Recsize x Blksize) x 2) + 15

5. NUMBUFS

One word containing either the word 'SINGLE' or the word 'DOUBLE'. Single indicates only one I/O buffer is desired for reading/writing data on mass storage. Double indicates two areas. If this parameter is omitted, single is assumed.

6. FILUSE

One word containing either the word 'INPUTb' or 'OUTPUT'. This parameter is only required if the NUMBUFS parameter equals 'DOUBLE', and it indicates whether the file is being read from or written to (it cannot be both) mass storage. Input files may be read in any order, but output files must be written sequentially. Examples:

```
            @ASG, T      MAS., F2//5000

COBOL       01 FILENAME   PICTURE X(12) VALUE 'MASbbb'.
            01 RECSIZE    PICTURE H9(10) VALUE 28.
            01 BLKSIZE    PICTURE H9(10) VALUE 10.
            01 BUFFER     PICTURE X(6) OCCURS 575 TIMES.
            01 NUMBUFS    PICTURE X(6) VALUE 'DOUBLE'.
            01 FILUSE     PICTURE X(6) VALUE 'OUTPUT'.

            ENTER FORTRAN OPENS SUBROUTINE
            REFERENCING FILENAME  RECSIZE  BLKSIZE
            BUFFER  NUMBUFS  FILUSE.

FORTRAN     DIMENSION IBUF(575),  IFILE (2)
            IFILE(1)  =  'MASbbb'
            IFILE(2)  =  'bbbbbb'
            IRECSZ    =  28
            IBLKSZ    =  10
            IBUFS     =  'DOUBLE'
            IUSE      =  'OUTPUT'
            CALL OPENS (IFILE, IRECSZ, IBLKSZ, IBUFX,
            IBUF, IUSE)
```

7.2.2 READS Entry Point

This entry point reads a specified record from mass storage and transfers the data to the user's program.

DOD AND FD COBOL

ENTER FORTRAN READS SUBROUTINE REFERENCING
FILENAME RECNO LOCATION EOF.

ASCII COBOL

CALL 'READS' USING FILENAME RECNO LOCATION EOF.

FORTRAN V

CALL READS (FILENAME RECNO LOCATION EOF)

1. FILENAME

   Two words containing a one-to-twelve fieldata character(s) filename. left-justified, and space-filled to the right.

2. RECNO

   One word containing a binary number specifying the number of the record to be read. IOPKG uses this number to calculate the location of the record on the mass storage device.

3. LOCATION

   A record area in the user's program where the data read from mass storage is to be placed.

4. EOF

   One word in the user's program. Set to binary 1 by IOPKG to indicate an end-of-file record was found while attempting to move the RECNO requested to the user. Set to binary 0 if RECNO requested was not an EOF record. Example:

```
COBOL      01  FILENAME  PICTURE X(12) VALUE 'MASbbb'.
           01  RECNO     PICTURE H9(10) VALUE 0.
           01  EOF       PICTURE H9(10) VALUE 0.
           01  REC-HOLD.
               02  NAME PICTURE X(06).
               02  REST  PICTURE X(06) OCCURS 27 TIMES.

           ADD 1 TO RECNO.
           ENTER FORTRAN READS SUBROUTINE
           REFERENCING FILENAME RECNO REC-HOLD EOF.
           IF EOF EQUALS 1 GO TO EOF-SITUATION.

FORTRAN   DIMENSION ILOC (28), IFILE (2)
           IFILE (1)  =  'MASbbb'
           IFILE (2)  =  'bbbbbb'
           IRECNO     =  IRECNO + 1
           CALL READS (IFILE, IRECNO, ILOC, IEOF)
           IF (IEOF.  EQ. 1) GO TO 100
```

7.2.3 WRITES Entry Point

This entry point transfers a data record from the user's program and writes it to mass storage.

DOD AND FD COBOL

```
ENTER FORTRAN WRITES SUBROUTINE REFERENCING
FILENAME RECNO LOCATION.
```

ASCII COBOL

```
CALL 'WRITES' USING FILENAME RECNO LOCATION.
```

FORTRAN V

```
CALL WRITES (FILENAME RECNO LOCATION)
```

1. FILENAME

Two words containing a one-to-twelve fieldata character(s) filename, left-justified, and space-filled to the right.

2. RECNO

One word containing a binary number specifying the number of the record to be written. IOPKG converts this number into an address on the mass storage device. Output files are not required to start writing with record number 1, but if the NUMBUFS parameter equals 'DOUBLE', each subsequent write command must reference a RECNO that is greater than the previous write command RECNO by 1. The user's program is responsible for incrementing RECNO before each WRITE command.

3. LOCATION

A record area in the user's program where data to be written to mass storage can be found. Examples:

```
COBOL      01 FILENAME  PICTURE X(12) VALUE 'MASbbb'.
           01 RECNO     PICTURE H9(10) VALUE 0.
           01 REC-HOLD.
              02 NAME   PICTURE X(06).
              02 REST   PICTURE X(06) OCCURS 27 TIMES.
                  .
           ADD 1 TO RECNO.
           ENTER FORTRAN WRITES SUBROUTINE
           REFERENCING FILENAME RECNO REC-HOLD.
```

```
FORTRAN        DIMENSION ILOC (28), IFILE (2)
               IFILE (1)  =  'MASbbb'
               IFILE (2)  =  'bbbbbb'
               IRECNO     =  IRECNO + 1
               CALL WRITES (IFILE, IRECNO, ILOC)
```

7.2.4 CLOSEI and CLOSEM Entry Points

The CLOSEI and CLOSEM entry points are used to close a mass storage file when processing on that file is completed. The CLOSEI entry point is used to close all input files, and the CLOSEM entry point is used to close output files. The CLOSEM entry point causes the final block of records (which may still be in memory) to be written to the mass storage device followed by a record containing a software end-of-file indicator. For this reason, only sequentially written output files should be closed with the CLOSEM routine; randomly updated input files should be closed with CLOSEI.

DOD AND FD COBOL

$$\text{ENTER FORTRAN} \left\{ \begin{matrix} \text{CLOSEI} \\ \text{CLOSEM} \end{matrix} \right\} \text{SUBROUTINE REFERENCING FILENAME.}$$

ASCII COBOL

$$\text{CALL} \left\{ \begin{matrix} \text{'CLOSEI'} \\ \text{'CLOSEM'} \end{matrix} \right\} \text{USING FILENAME.}$$

FORTRAN V

$$\text{CALL} \left\{ \begin{matrix} \text{CLOSEI} \\ \text{CLOSEM} \end{matrix} \right\} \text{(FILENAME)}$$

FILENAME

Two words containing a one-to-twelve fieldata character(s) filename, left-justified, and space-filled to the right. Examples:

```
COBOL       01  FILENAME  PICTURE X(12) VALUE 'MASbbb'.
            01  RECNO     PICTURE H9(10) VALUE 0.
            01  REC-HOLD
                02  NAME  PICTURE X(06).
                02  REST  PICTURE X(06) OCCURS 27 TIMES

            ENTER FORTRAN CLOSEM SUBROUTINE
            REFERENCING FILENAME

FORTRAN     DIMENSION ILOC (28), IFILE (2)
            IFILE (1)  =  'MASbbb'
            IFILE (2)  =  'bbbbbb'
            CALL CLOSEM (IFILE).
```

7.2.5 IOPKG Error Messages

IOPKG automatically generates error messages and terminates execution of the users program when fatal error conditions are detected. The format for diagnostic messages created by IOPKG is as follows:

```
IOPKG ERROR  nn  FUNCTION  n
FILE NAME = xxxxxxxxxxxx
```

The error codes returned by IOPKG may be found in the EXEC VIII Programmers Reference Manual (UP-4144, Revison 3), Appendix C, Page C-17. Error and function codes returned by IOPKG and not documented in the Programmers Reference Manual are listed below.

Error Code	Meaning
05	ATTEMPT TO READ FROM AN UNASSIGNED AREA OF MASS STORAGE.
51	FILE ALREADY OPEN.
52	EOF AGRUMENT NOT SPECIFIED.
53	IOPKG BUFFER IS IN USE.
54	EXCEEDED MAXIMUM NUMBER OF OPENED FILES (20).
55	FILE NOT OPENED.
56	RECORD SIZE LESS THAN OR EQUAL TO 0.
57	BLOCK SIZE LESS THAN OR EQUAL TO 0.
58	DOUBLE BUFFER USAGE NOT INPUT OR OUTPUT.
59	READ COMMAND ISSUED TO OUTPUT FILE.
60	NOT USED.
61	RECORD NUMBER LESS THAN OR EQUAL TO 0.
62	ATTEMPT TO READ BEYOND EOF.
63	NEW OUTPUT RECNO NOT EQUAL OLD OUTPUT RECNO PLUS 1.
64	NUMBER OF BUFFERS NOT EQUAL SINGLE OR DOUBLE.
65	INHIBIT READ INVALID WITH SINGLE BUFFER.

Function Code	Meaning
0	ERROR ON CALL TO OPENS ENTRY POINT
1	ERROR ON CALL TO READS ENTRY POINT
2	ERROR ON CALL TO WRITES ENTRY POINT
3	ERROR ON CALL TO CLOSEM ENTRY POINT
4	ERROR ON CALL TO CLOSEI ENTRY POINT

7.3 MREAD CARD READER SUBROUTINE

MREAD is a UNIVAC 1108 Assembler Language subroutine which may be used for reading card reader files. Each call to the MREAD subroutine causes one card to be read from the card reader and transferred to a buffer in the user's program. The calling sequence for MREAD is as follows:

DOD AND FD COBOL

ENTER FORTRAN MREAD SUBROUTINE REFERENCING BUFFER EOF.

ASCII COBOL

CALL 'MREAD' USING BUFFER EOF.

FORTRAN V

CALL MREAD (BUFFER, EOF).

1. BUFFER

   The core area or buffer in the user's program where data read from the card reader is to be placed. This buffer must be 84 characters or 14 words in length.

2. EOF

   One word containing a binary number used as an end-of-file switch. This switch will contain a value of zero when an EOF condition is reached; otherwise, it will contain the value for the number of words read and moved into BUFFER.

   Examples:

```
COBOL       01  EOF        PICTURE H9(10) VALUE 0.
            01  REC-HOLD.
                02 REST   PICTURE X(06) OCCURES 14 TIMES.

            ENTER FORTRAN MREAD SUBROUTINE
            REFERENCING REC-HOLD EOF.  IF EOF
            EQUALS 0 GO TO EOF-SITUATION.

FORTRAN     DIMENSION ILOC (14)
            CALL MREAD (ILOC, IEOF)
            IF (IEOF . EQ. 0) TO 100
```

7.4 MPRINT PRINTER SUBROUTINE

MPRINT is a UNIVAC 1108 Assembler Language subroutine which may be used for writing output reports to a printer. Each call to the MPRINT subroutine causes one record to be transferred from the user's buffer and written to the printer. The calling sequence for MPRINT is as follows:

DOD AND FD COBOL

```
ENTER FORTRAN MPRINT SUBROUTINE REFERENCING BUFFER
NUMBWDS CARRIAGE.
```

ASCII COBOL

```
CALL 'MPRINT' USING BUFFER NUMBWDS CARRIAGE.
```

FORTRAN V

```
CALL MPRINT (BUFFER, NUMBWDS, CARRIAGE).
```

1. BUFFER

   The core area or buffer in the user's programs from which a record or data is to be written.

2. NUMBWDS

   One word containing a binary number representing the number of words that are to be printed in the print line (normally 22 words per line).

3. CARRIAGE

   One word containing a binary number used to control the carriage for the printer.

0	=	No Spacing
1	=	Single Space
2	=	Double Space
63	=	Page Eject
N	=	N Space

Examples:

```
COBOL       01  NUMBWDS      PICTURE H9(10) VALUE 22.
            01  CARRIAGE     PICTURE H9(10) VALUE 1.
            01  PRINT-LINE   PICTURE X(132).

            ENTER FORTRAN MPRINT SUBROUTINE
            REFERENCING PRINT-LINE NUMBWDS CARRIAGE.

FORTRAN     DIMENSION IPRINT (22)
            ICARR    =  1
            INUMB    =  22
            CALL MPRINT (IPRINT, INUMB, ICARR)
```

7.5 MPRINA ALTERNATE PRINTER SUBROUTINE

MPRINA is a UNIVAC 1108 Assembler Language subroutine which may be used for writing output reports to an alternate print file. Each call to the MPRINA subroutine causes one record to be transferred from the user's buffer and written to the alternate print file. The alternate print file must be closed by the user after all records have been written. This may be done by dynamically sending a @BRKPT filename image to the Executive Operating System from within the executing program (via some type of subroutine using the Executive Request CSF$), or by the use of the @BRKPT Executive Control card after completion of program execution. The calling sequence for MPRINA is as follows:

DOD AND FD COBOL

ENTER FORTRAN MPRINA SUBROUTINE REFERENCING FILENAME BUFFER NUMBWDS CARRIAGE.

ASCII COBOL

CALL 'MPRINA' USING FILENAME BUFFER NUMBWDS CARRIAGE.

FORTRAN V

CALL MPRINA (FILENAME, BUFFER, NUMBWDS, CARRIAGE).

1. FILENAME

   Two words containing a one-to-nine fieldata character(s) filename, left-justified, and space-filled to the right.

2. BUFFER

   The core area or buffer in the user's programs from which a record or data is to be written.

3. NUMBWDS

   One word containing a binary number representing the number of words that are to be printed in the print line (normally 22 words per line).

4. CARRIAGE

   One word containing a binary nun ber used to control the carriage for the printer.

0	=	No Spacing
1	=	Single Space
2	=	Double Space
63	=	Page Eject
N	=	N Space

Examples:

```
COBOL       01  BRK-FILE       PICTURE X(18) VALUE
                                 '@BRKPT PRINT$ .  '
            01  FILENAME       PICTURE X(12) VALUE
                                 'ALTPRINT'.
            01  NUMBWDS        PICTURE H9(10) VALUE 22.
            01  CARRIAGE       PICTURE H9(10) VALUE 1.
            01  PRINT-LINE     PICTURE X(132).

            ENTER FORTRAN MPRINA SUBROUTINE
            REFERENCING FILENAME PRINT-LINE NUMBWDS
            CARRIAGE.
            .
            .
            .
            ENTER FORTRAN CSFASG SUBROUTINE
            REFERENCING BRK-FILE.

FORTRAN     DIMENSION ILOC(22), IUNIT(2), IBRK(3)
            DATA IUNIT/'ALTPRINT    '/
            IBRK/'@BRKPT PRINT$ .  '/
            ICARR  =  1
            INWDS  =  22
            CALL MPRINA (IUNIT, ILOC, INWDS, ICARR)
            .
            .
            .
            CALL CSFASG (IBRK)
```

7.6 ROLLOUT UNLOAD PROGRAM

The ROLLOUT program is used to create a magnetic tape backup of the MIRADS User's File Set from mass storage. The magnetic tape must be assigned as UFSTAP. The files are written to tape in UNIVAC 1108 ROLLOUT format in the following order: (DIC, SAV, MAS, DRL, and IND).

The commands for copying the MIRADS User's File Set from mass storage to tape are:

```
@ASG,T  UFSTAP.,8C,SAVEnn . MIRADS User's File Set Tape
@XQT    MIR*ADS.ROLLOUT
```

When the program responds with the message

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

The user may enter either the word

STANDARD

or

A 1-to-12 character qualifier (optional) followed by a one-to-nine character filename (required) followed by a cycle number (optional). If present, the qualifier must be separated from the filename by an asterisk, and the cycle number must be enclosed in parenthesis.

If the word STANDARD is entered, the following actions are taken by the program:

1. The DIC, SAV, MAS, DRL, and IND files are copied from mass storage to the UFSTAP tape in UNIVAC 1108 ROLLOUT format through the use of the U-1108 FURPUR COPY,GM command.
2. An End-of-File mark is written on the tape after each file.

The user may now free the UFSTAP output tape. The loading process for temporary files is complete.

If the user enters a QUALIFIER*FILENAME (CYCLE), the following actions are taken by the program:

1. The DICfilename, SAVfilename, MASfilename, DRLfilename, and INDfilename files (with appropriate qualifier and cycle number) are copied from mass storage to the UFSTAP in UNIVAC 1108 ROLLOUT format through the use of the U-1108 FURPUR COPY, GM command.
2. An End-of-File mark is written on the tape after each file.

The user may now free the USFSTAP output tape. The loading process for cataloged files is complete.

7.7 ROLLIN LOAD PROGRAM

The ROLLIN program is used to load the MIRADS User's File set from magnetic tape to a mass storage device. The magnetic tape must be assigned as UFSTAP. The files must be in UNIVAC 1108 ROLLOUT format and in the following order (DIC, SAV, MAS, DRL, and IND). The files will be in the prescribed format and order when they have been created by the MIRADS ROLLOUT program which is documented in Paragraph 7.6. The commands for copying the MIRADS User's File Set from tape to mass storage are:

```
@ASG,T  UFSTAP.,8C,REEL NUMBER . MIRADS User's File Set Tape
@XQT    MIR*ADS.ROLLIN
```

When the program responds with the message

```
ENTER QUALIFIER*FILENAME(CYCLE)
```

The user may enter either the word

STANDARD

or

A 1-to-12 character qualifier (optional) followed by a one-to-nine character filename (required) followed by a cycle number (optional). If present, the qualifier must be separated from the filename by an asterisk, and the cycle number must be enclosed in parenthesis.

If the word STANDARD is entered, the following actions are taken by the program:

1. The DIC, SAV, MAS, DRL, and IND files are assigned as temporary files on mass storage.
2. The input files are copied or rolled-in from tape into the temporary files on mass storage through the use of the U-1108 FURPUR COPY,G command.

The user may now free the UFSTAP input tape. The loading process for temporary files is complete.

If the user enters a QUALIFIER*FILENAME (CYCLE), the following actions are taken by the program:

1. The DICfilename, SAVfilename, MASfilename, DRLfilename, and INDfilename files are cataloged and assigned with the appropriate qualifier and cycle number on mass storage.
2. The input files are copied or rolled-in from tape into the cataloged files on mass storage through the use of the U-1108 FURPUR COPY, G command.
3. The cataloged files are then FREE'd from the run to ensure their being entered into the UNIVAC 1108 Executive's Master File Dictionary.

The user may now free the UFSTAP input tape. The loading process for cataloged files is complete.

7.8. DUMP PROGRAM

The DUMP program is a generalized dump program used to dump mass storage data files created by the MIRADS IOPKG. The size of the print line is 128 positions. The following command must be entered to execute the program:

```
@XQT      MIR*ADS.DUMP
```

Upon execution, the program requests the user to enter the parameters.

```
ENTER (FILENAME, RECSIZE, BLKSIZE, RECNUMB,
        FORMAT, NUMBRECS)
```

1. FILENAME

   Name of the file to be dumped.

2. RECSIZE

   Number of words in each record.

3. BLKSIZE

   Number of records in each block.

4. RECNUMB

   Record number of the first record to be dumped relative to the start of the file. The first record is record number 1.

5. FORMAT

   This is the alphabetic character specifying the dump format.

A	-	Alphabetic Dump
O	-	Octal Dump
OA	-	Alphabetic and Octal Dump
AO	-	Alphabetic and Octal Dump
I	-	Convert each Word from Binary to its Decimal Equivalent

6. NUMBRECS

   The number of records to be dumped.

The parameters RECSIZE and BLKSIZE may be varied by the user for his benefit; however, caution must be used because the program may not be able to detect the software end-of-file if they are not the same as used for file creation. This feature enables a user to selectively dump small sections of a large file without the necessity of dumping the entire file.

After dumping the requested number of records, the program will again request input from the user.

```
ENTER (STOP OR NF OR RECNUMB, FORMAT, NUMBRECS)
```

1. STOP

   This will terminate the dump program.

2. NF

   Requests a dump on a new file and will cause the program to recycle to the beginning for new parameters.

3. RECNUMB, FORMAT, and NUMBRECS)

   These parameters are to be input if more of the initial file is to be dumped.

When a software end-of-file is encountered, execution of the program is automatically terminated.

If the program is being executed in batch mode, cards containing the parameters must follow the @XQT card in the correct order for execution.

The following is an example of a batch RUN executing AMDUMP:

```
@RUN
.
.
.
@ASG, T             FILEA., F2/8/TRK/64
@ASG, T             FILEB., F2/1/POS/1
.
.
.
@XQT                MIR*ADS.DUMP
FILEA,28,8,1,A,2
173,I,1
NF
FILEB,128,4,8,O,5
STOP
.
.
@FIN
```

The first and second records of FILEA will be dumped in an alphabetic format as the result of the first parameter command to the DUMP program. The second command will dump record number 173 of FILEA converting each word from binary to its decimal equivalent. The third command will direct the dumping of FILEB. Beginning with record number 8 of FILEB, five records will be dumped in an octal format. Each record contains 128 words. The last command will terminate the dump program.

An on-line execution of the dump program will be identical to the batch RUN. Each input command is processed as it is entered and the

messages calling for entry of data will print after the execution of the previous entry.

7.9 DICTOCARD PROGRAM

The Dictionary-to-Card program is used to convert the MIRADS Dictionary file (DIC) back to card image format. The cards may then be easily modified by using the UNIVAC 1108 Text Editor program. The card images are written in a temporary UNIVAC 1108 program file named CRDFILE with elementname named cards. The file, CRDFILE, is assigned automatically by the program. The commands for converting the DIC file to card format are:

```
@ASG, A     DICfilename
@USE        DIC, DICfilename
@XQT        MIR*ADS. DICTOCARD
@FREE       DIC
```

The cards may then be updated using the UNIVAC 1108 Text Editor program as follows:

```
@ED, U      CRDFILE. CARDS
   .
   .
   .
Text Editor commands
   .
   .
   .
EXIT
```

After the cards have been modified using the Text Editor, the updated file may be used as card image input to the MIRADS

Dictionary generation program (DICGEN) through the use of the following UNIVAC 1108 control card:

```
@ADD      CRDFILE. CARDS
```

SECTION 8 - HOW TO LOAD THE MIRADS LIBRARY

8.1 INTRODUCTION

The MIRADS Systems Release Package contains a seven track magnetic tape (800BPI Odd Parity) with the following seven files written in UNIVAC 1108 ROLLOUT format (@COPY, GM):

1. MIRADS Library
2. All MIRADS Symbolic, Relocatable, and Absolute Elements
3. DICNATIONS
4. SAVNATIONS
5. MASNATIONS NATIONS User's File Set
6. DRLNATIONS
7. INDNATIONS

An end-of-file mark is written after each file on the tape.

8.2 SAMPLE RUN STREAM

The sample run stream below may be used to load the MIRADS Library:

```
@RUN
@ASG, T         LIBTAP, 8C, REEL NUMBER
@ASG, UPRV      MIR*ADS(+1)., F2/1/POS/5
@COPY, G        LIBTAP., MIR*ADS(+1).
@FREE           MIR*ADS(+1).
```

The UPRV options in the above ASG command have the following effects:

U Catalog this file when a FREE command is issued or the RUN terminates, whether there has been an error in the run stream or not.

P This is a public file and may be accessed by computer runs using different Project-ID's in the run card.

R Catalog this file as a read-only file.

V Keep this file mass storage resident; do not roll it out to tape.

The qualifier and filename, MIR*ADS, must be used for cataloging the MIRADS Library because they are referenced in this way throughout the entire MIRADS System.

If the user wishes to run a sample query using the NATIONS User's File Set, the following additional commands are required:

```
@MOVE         LIBTAP.,1
@USE          UFSTAP, LIBTAP
@XQT, R       MIR*ADS. LOADER
NATIONS
@FREE         LIBTAP.
@XQT          MIR*ADS. MIRADS
NATIONS
QUERY, CITY PRESENT.
SORT, STATE ASCENDING, CITY ASCENDING.
PRINT, COUNTRY GROUP 1, STATE GROUP 2, CITY.
RUN
ENTER A BLANK LINE
STOP
@FIN
```

Users may avoid the problem of having to recatalog the MIRADS Library periodically by using the 1108 SECURE processor feature as defined in

Chapter 19 of UNIVAC 1100 Series Operating System Programmer's Reference Manual (UP-4144 Revision 3). The SECURE processor protects the physical security of cataloged files which reside on mass storage by providing tape backups. If the MIRADS Library becomes unloaded for any reason, the features of SECURE can be used to reload the file.

APPENDIX A - SAMPLE MIRADS APPLICATIONS

A.1 A DATA BASE WITH ALL DATA TYPES

The MIRADS Library contains all the elements required to build a five-record Data Base containing fields of all the data types. The element BUILDTEST can be added to any user run stream to generate a Dictionary listing, build a temporary User's File Set, and process two queries. The queries provide the user with visibility of the contents of most of the fields of the Data Base. Building the Data Base and processing the queries requires less than five minutes elapsed time and approximately two seconds of CPU time. The run stream of Figure A-1 is processed to build the Dictionary of Figure A-2 with the command.

```
@ADD      MIR*ADS.BUILDTEST
```

```
MIR*ADS.BUILDTEST
        @XQT MIR*ADS.TESTGEN
        @ADD MIR*ADS.TESTDATA
        @XQT,SU MIR*ADS.DICGEN
        @ADD MIR*ADS.TESTDIC
        @XQT MIR*ADS.DRLGEN
        @XQT MIR*ADS.INDGEN
        @XQT MIR*ADS.MIRADS
        STANDARD
        QUERY,FDA PRESENT.
        SORT,FDSD DESCENDING.
        PRINT,FDA,FDAN,FDN,FDS,FDSD,B,BP,FPDP,FPSP.
        SAVE,SAVE-QUERY1
        RUN

        Q,FDA P.
        P,FDSD,PPOS 80 FDTLU,FDTLU LOOKUP.
        SAVE,SAVE-QUERY2
        RUN

        LIST
```

Figure A-1. Listing of Element BUILDTEST

```
   06 02 75                MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING          PAGE   1

  FILE NAME = TEST             RECORD SIZE (WDS) =    28    BLOCK SIZE =    64    LEVELS =   1    FILE SECURITY =

          DICTIONARY RECORD TYPE         RECORD IDENTIFIER        START LOCATION      END LOCATION       RECORD SIZE (WDS)

                  101                          ///                      0000                0000                0028

    FIELD      FIELD DICT  LOCATION   FIELD I  UP              REPORT FIELD TITLE              TITLE  DEC DATA SRCH     TABLE LOOKUP
    NAME        NO   LVL  START END   SIZE N DATE                                              CHARS  I O TYPE TYPE  NO  WIDTH  GLOBAL
                     CODE                  D                                                          N                  (MAX)

  B            0010  101  0001 0006   006  Y   Y   BINARY                                        06   0 0  07   RG
  BN           0020  101  0007 0012   006  Y   Y   NEG-BINARY                                    10   0 0  07   RG
  BP           0030  101  0013 0018   006  Y   Y   POS-BINARY                                    10   0 0  07   RG
  FDA          0040  101  0019 0024   006  Y   Y   FD-ALPHA                                      08   0 0  00   RG
  FDAN         0050  101  0025 0030   006      Y   FD-AN                                         05   0 0  01   RG
  FDN          0060  101  0031 0039   009  Y   Y   FD-NUMERIC                                    10   0 0  02   RG
  FDS          0070  101  0040 0048   009  Y   Y   FD-SIGNED                                     09   1 1  03   RG
  FDSD         0080  101  0049 0060   012  Y   Y   FD-SIGNED-DECIMAL                             17   2 2  05   RG
  FDTLU        0090  101  0133 0138   006  Y   Y   LOOKUP-FIELD                                  12   0 0  01   RG   001   10
  FDUD         0100  101  0061 0072   012  Y   Y   FD-UNSIGNED-DECIMAL                           19   3 3  04   RG
  FD60         0110  101  0073 0132   060          60-CHARACTER-NARRATIVE                        22   0 0  01   KP
  FPDP         0120  101  0139 0150   012  Y   Y   DOUBLE-PRECISION                              16   0 4  09   RG
  FPSP         0130  101  0151 0156   006  Y   Y   SINGLE-PRECISION                              16   0 3  08   RG
  KF           0140  101  0073 0078   006  Y       KEY                                           03   0 0  01   RG


 06 02 75                MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING          PAGE   2

FILE NAME = TEST             RECORD SIZE (WDS) =    28    BLOCK SIZE =    64    LEVELS =   1    FILE SECURITY =

      TABLE NO      DATABASE VALUE     REPORT TITLE

        001                            *NO ENTRY*
        001            000001          PROGRAMMER
        001            000002          ANALYST
        001            000003          SCIENTIST
        001            000005
```

Figure A-2. Test Dictionary Created by Element BUILDTEST

A.2 A SINGLE LEVEL DATA BASE

Figures A-3 through A-5 illustrates the data cards, generation program and coded Dictionary of a simple single level application. Figure A-6 illustrates the Dictionary listing produced by the Dictionary Generator Program (DICGEN).

```
000001JONES                       S  C20113 [redacted] 225  JONES  VALLEY   01012027000M06
000002BURNS                       W  D32080            1442  NORTH  BELAIR  01011023500M04
000005KING                        W  L38120            3511  W  GEORGIA  AVE04034014400M05
000009ZORNASKI                    P  U36042            4244  JEAN  ROAD     05043015000F01
000011THOMPSON                    J  K45061            504  DOWNING  DRIVE  01011009600M01
```

Figure A-3. Single Level Application Data Cards

1100 ASCII COBOL SOURCE LISTING

```
IDENTIFICATION DIVISION.
PROGRAM-ID. DBGEN1.
REMARKS.    THIS PROGRAM READS AN EXISTING CARD FILE TO
     GENERATE A MIRADS MASTER FILE.  THE MIRADS MASTER FILE
     WILL BE STRUCTURED WITH NO LEVELS OF FILE SUBORDINATION
     AND A SINGLE RECORD TYPE.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER. UNIVAC-1108.
OBJECT-COMPUTER. UNIVAC-1108.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
     SELECT CARD-FILE ASSIGN TO CARD-READER.
DATA DIVISION.
FILE SECTION.
FD CARD-FILE
     LABEL RECORDS ARE OMITTED
     DATA RECORD IS CARD-RECORD.
01   CARD-RECORD                               PICTURE X(84).
WORKING-STORAGE SECTION.
01   MAS-UNIT      VALUE  'MAS          '      PICTURE X(12).
01   MAS-REC-SIZE  VALUE  14                   PICTURE H9(10).
01   MAS-BLK-SIZE  VALUE  128                  PICTURE H9(10).
01   MAS-REC-NBR   VALUE  0                    PICTURE H9(10).
01   MAS-RECORD.
     02 MAS-REC    OCCURS 14                   PICTURE X(6).
01   MAS-BUFFER.
     02 MAS-BUF    OCCURS 1807                 PICTURE X(6).
PROCEDURE DIVISION.
MASGEN1-OPEN-FILES.
     OPEN INPUT CARD-FILE.
     CALL 'OPENS' USING   MAS-UNIT       MAS-REC-SIZE
                          MAS-BLK-SIZE   MAS-BUFFER.
CARD-READ-IOPKG-WRITE-LOOP.
     READ CARD-FILE INTO MAS-RECORD AT END GO TO CLOSE-FILES.
     ADD 1 TO MAS-REC-NBR.
     CALL 'WRITES' USING MAS-UNIT MAS-REC-NBR MAS-RECORD.
     GO TO CARD-READ-IOPKG-WRITE-LOOP.
CLOSE-FILES.
     CLOSE CARD-FILE.
     CALL 'CLCSEM' USING MAS-UNIT.
     STOP RUN.
```

Figure A-4. Single Level Data Base Generation Program

NAME PERSØNNEL DICTIØNARY		
ACCOUNT	PHONE	

FILE NAME CARD

SHEET 1 OF 3

DATE

1 ACT	2 TYPE	3 FILE NAME	12 NBR DATA BASE RECORDS	20 MAX REC SIZE	24 RECS PER BLOCK	28 NBR LEVEL	30 SECURITY KEY	38 BLANK
I	F	PERSØNNEL	2000	14	128	1		

PASSWORD CARD

1 ACT	2 TYPE	3 PASSWORD	15 UPIND	16 BLANK
I	I	ANLYST-VM	Y	
I	I	ACCTG-AR	Y	
I	I	PERSONNEL-PL	N	
I	I	CLERK-BT	N	
	I			
	I			
	I			
	I			
	I			
	I			

RECORD IDENTIFIER CARD

1 ACT	2 TYPE	3 REC TYPE	6 REC ID	9 START LOC	13 END LOC	17 REC SIZE	21 BLANK
I	L	101	///			14	
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						

MSFC - Form 432-2 (Rev. May 1975)

Figure A-5. Single Level Data Base Dictionary Cards

NAME PERSØNNEL DICTIØNARY	
ACCOUNT	PHONE

FIELD DEFINITION CARD 1

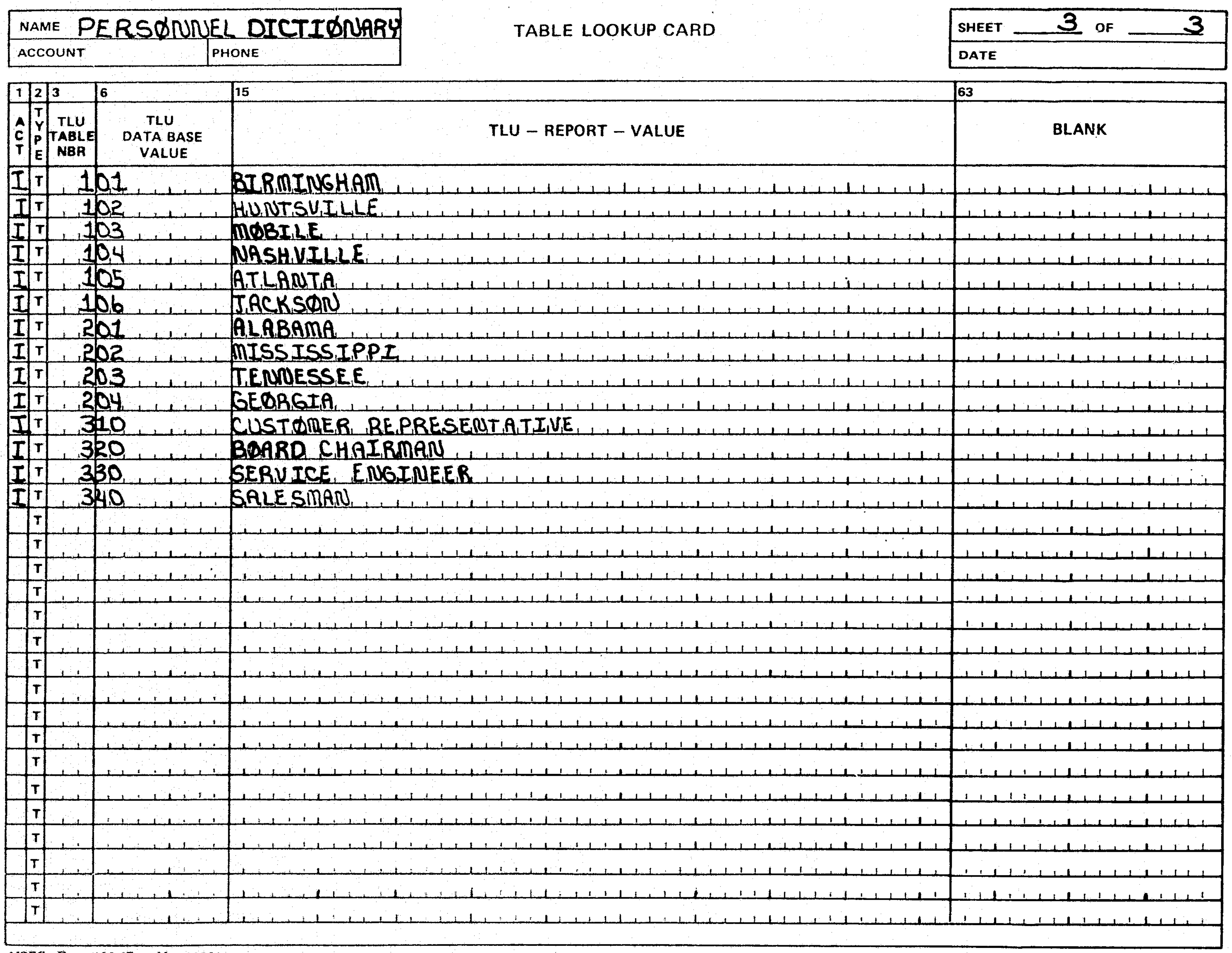
SHEET 2 OF 3

DATE

1	2	3 6	7	8	20	56	59	63	67	69	71	72	73	75	76	77	80
ACT	TYPE	FIELD NBR	NBR	FIELD NAME	REPORT TITLE	REC TYPE	START LOC	END LOC	NBR OCC	SRCH TYPE	INDEX	UP IND	DATA TYPE	DECS IN	DECS OUT	TLU TABLE NBR	GTLU
I	M	10	1	EMP	EMPLØYEE NUMBER	101	1	6			Y	N	2				
I	M	20	1	NAME	EMPLØYEE NAME	101	7	30		CH	Y	Y	1				
I	M	30	1	DØB	DATE ØF BIRTH (YRMODA)	101	31	36			N	Y	2				
I	M	40	1	YRDØB	YEAR ØF BIRTH	101	31	32			N	Y	2				
I	M	50	1	MØDØB	MØNTH ØF BIRTH	101	33	34			N	Y	2				
I	M	60	1	DADØB	DAY ØF BIRTH	101	35	36			N	Y	2				
I	M	70	1	SSN	SØCIAL SECURITY NUMBER	101	37	47			N	Y	1				
I	M	80	1	STREET	STREET ADDRESS	101	48	65			N	Y	1				
I	M	90	1	CITY	CITY	101	66	67			N	Y	2			1	
I	M	100	1	STATE	STATE	101	68	69			N	Y	2			2	
I	M	110	1	JØB	JØB TITLE	101	70	71			Y	Y	2			3	
I	M	120	1	PAY	SALARY	101	72	76			N	N	2				
I	M	130	1	SEX	SEX	101	77	77			N	Y	0				
I	M	140	1	DEP	DEPENDENTS	101	78	79			N	Y	2				
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														

MSFC - Form 433-4 (Rev. May 1976)

Figure A-5. Single Level Data Base Dictionary Cards (Continued)



NAME PERSØNNEL DICTIØNARY

ACCOUNT | PHONE

TABLE LOOKUP CARD

SHEET 3 OF 3

DATE

1 ACT	2 TYPE	3 TLU TABLE NBR	6 TLU DATA BASE VALUE	15 TLU – REPORT – VALUE	63 BLANK
I	T	1	01	BIRMINGHAM	
I	T	1	02	HUNTSVILLE	
I	T	1	03	MØBILE	
I	T	1	04	NASHVILLE	
I	T	1	05	ATLANTA	
I	T	1	06	JACKSØN	
I	T	2	01	ALABAMA	
I	T	2	02	MISSISSIPPI	
I	T	2	03	TENNESSEE	
I	T	2	04	GEØRGIA	
I	T	3	10	CUSTØMER REPRESENTATIVE	
I	T	3	20	BØARD CHAIRMAN	
I	T	3	30	SERVICE ENGINEER	
I	T	3	40	SALESMAN	
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				

MSFC - Form 432 (Rev. May 1975)

Figure A-5. Single Level Data Base Dictionary Cards (Continued)

```
 06 02 75              MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING          PAGE   1
FILE NAME = PERSONNEL        RECORD SIZE (WDS) =    14    BLOCK SIZE =   128    LEVELS =   1    FILE SECURITY =


                                        INITIATOR PASSWORD   UPDATE INDICATOR

                                           ACCTG-AR                Y
                                           ANLYST-VH               Y
                                           CLERK-BJ                N
                                           PERSONNEL-PL            N


    06 02 75              MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING          PAGE   1
   FILE NAME = PERSONNEL        RECORD SIZE (WDS) =    14    BLOCK SIZE =   128    LEVELS =   1    FILE SECURITY =
        DICTIONARY RECORD TYPE        RECORD IDENTIFIER        START LOCATION        END LOCATION        RECORD SIZE (WDS)

                  101                       ///                    0000                 0000                   0014

   FIELD     FIELD DICT  LOCATION  FIELD I  UP               REPORT FIELD TITLE               TITLE  DEC DATA SRCH    TABLE LOOKUP
   NAME       NO   LVL  START END  SIZE N DATE                                                CHARS  I O TYPE TYPE  NO  WIDTH  GLOBAL
                   CODE                 D                                                            N                 (MAX)

EMP          0010  101  0001 0006  006  Y      EMPLOYEE NUMBER                                  15   0 0  02   RG
NAME         0020  101  0007 0030  024  Y   Y  EMPLOYEE NAME                                    13   0 0  01   CH
DOB          0030  101  0031 0036  006      Y  DATE OF BIRTH (YRMODA)                           22   0 0  02   RG
YRDOB        0040  101  0031 0032  002      Y  YEAR OF BIRTH                                    13   0 0  02   RG
MODOB        0050  101  0033 0034  002      Y  MONTH OF BIRTH                                   14   0 0  02   RG
DADOB        0060  101  0035 0036  002      Y  DAY OF BIRTH                                     12   0 0  02   RG
SSN          0070  101  0037 0047  011      Y  SOCIAL SECURITY NUMBER                           22   0 0  01   RG
STREET       0080  101  0048 0065  018      Y  STREET ADDRESS                                   14   0 0  01   RG
CITY         0090  101  0066 0067  002      Y  CITY                                             04   0 0  02   RG   001   10
STATE        0100  101  0068 0069  002      Y  STATE                                            05   0 0  02   RG   002   11
JOB          0110  101  0070 0071  002  Y   Y  JOB TITLE                                        09   0 0  02   RG   003   23
PAY          0120  101  0072 0076  005         SALARY                                           06   0 0  02   RG
SEX          0130  101  0077 0077  001      Y  SEX                                              03   0 0  00   RG
DEP          0140  101  0078 0079  002      Y  DEPENDENTS                                       10   0 0  02   RG
```

Figure A-6. Single Level Data Base Dictionary Listing

```
06 02 75            MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING          PAGE   2
FILE NAME = PERSONNEL       RECORD SIZE (WDS) =    14    BLOCK SIZE =   128    LEVELS =   1    FILE SECURITY =

     TABLE NO      DATABASE VALUE      REPORT TITLE

        001              01               BIRMINGHAM
        001              02               MOBILE
        001              03               HUNTSVILLE
        001              04               NASHVILLE
        001              05               ATLANTA
        001              06               JACKSON
        002              01               ALABAMA
        002              02               MISSISSIPPI
        002              03               TENNESSEE
        002              04               GEORGIA
        003              10               CUSTOMER REPRESENTATIVE
        003              20               BOARD CHAIRMAN
        003              30               SERVICE ENGINEER
        003              40               SALESMAN
```

Figure A-6. Single Level Data Base Dictionary Listing (Continued)

A.3 A MULTI-LEVEL DATA BASE

Figures A-7 through A-9 illustrate the data cards, generation program and coded Dictionary of a multi-level NATIONS application. Figure A-10 illustrates the Dictionary listing produced by the Dictionary Generator Program (DICGEN). This NATIONS application is the User's File Set included on the system Release Tape of all MIRADS System Releases.

```
01  A   UNITED STATES OF AMERICAFORD, GERALD R.           203235298 36086723
0101B   ALABAMA       MONTGOMERY      3444165 516902129WALLACE,GEORGE C
0101C001AUTAUGA       PRATTVILLE        24460    599
0101C002BALDWIN       MAY MINETTE       59362   1578
0101C003BARBOUR       CLAYTON           22543    891
0101C037JEFFERSON     BIRMINGHAM       644991   1115
0101C045MADISON       HUNTSVILLE       186540    803
0103B   CALIFORNIA    SACRAMENTO     19953134158693010BROWN, EDMUND G JR
0103C001ALAMEDA       OAKLAND         1073184    733
0103C019LOS ANGELES   LOS ANGELES     7036887   4069
0151B   WYOMING       CHEYENNE         332416 979145009HATHAWAY,STANLEY K
0151C023WESTON        NEWCASTLE          6307   2407
```

Figure A-7. Multi-Level Application Data Cards

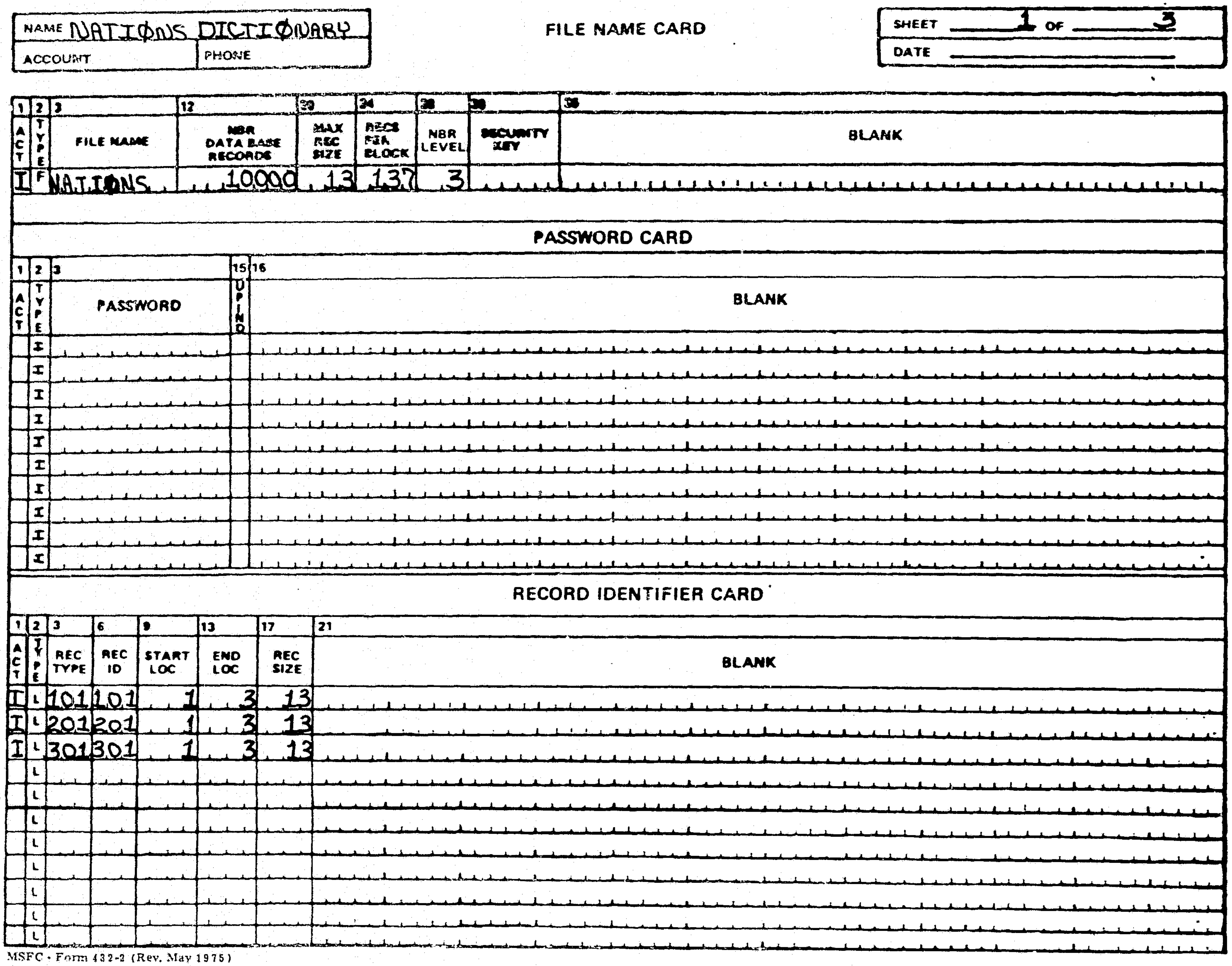
1100 ASCII COBOL SOURCE LISTING

```
IDENTIFICATION DIVISION.
PROGRAM-ID.  DBGEN2.
REMARKS.
           THIS PROGRAM GENERATES A MIRADS MASTER FILE FROM AN
    EXISTING CARD FILE.  THE MIRADS MASTER FILE WILL BE
    STRUCTURED WITH TWO LEVELS OF FILE SUBORDINATION.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  UNIVAC-1108.
OBJECT-COMPUTER.  UNIVAC-1108.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT CARD-FILE ASSIGN TO CARD-READER.
DATA DIVISION.
FILE SECTION.
FD CARD-FILE
    LABEL RECORDS ARE OMITTED
    DATA RECORDS ARE COUNTRY-CARD STATE-OR-COUNTY-CARD.
01  COUNTRY-CARD.
    02  FILLER                                  PICTURE X(4).
    02  CARD-TYPE                               PICTURE X.
    02  FILLER                                  PICTURE X(3).
    02  COUNTRY-CARD-DATA.
        03  COUNTRY-AND-PRESIDENT               PICTURE X(48).
        03  POPULATION-COUNTRY                  PICTURE X(10).
        03  AREA-COUNTRY                        PICTURE X(8).
        03  CONTINENT                           PICTURE 9(1).
    02  FILLER                                  PICTURE X(9).
01  STATE-OR-COUNTY-CARD.
    02  FILLER                                  PICTURE X(8).
    02  STATE-OR-COUNTY-DATA.
        03  STATE-OR-COUNTY                     PICTURE X(14).
        03  CAPITAL-OR-SEAT                     PICTURE X(16).
        03  POPULATION-STATE-COUNTY             PICTURE X(8).
        03  AREA-STATE-COUNTY                   PICTURE X(6).
        03  STATE-RANK-IN-POP                   PICTURE X(2).
        03  STATE-RANK-IN-AREA                  PICTURE X(2).
        03  STATE-GOVERNOR                      PICTURE X(24).
WORKING-STORAGE SECTION.
01  MAS-UNIT          VALUE   'MAS          '   PICTURE X(12).
01  MAS-REC-SIZE  VALUE   13                    PICTURE H9(10).
01  MAS-BLK-SIZE  VALUE   137                   PICTURE H9(10).
01  MAS-REC-NBR   VALUE 0                       PICTURE H9(10).
01  MAS-RECORD.
    02  MAS-REC-TYPE                            PICTURE 9(3).
    02  FILLER    VALUE SPACES                  PICTURE X(3).
    02  MAS-REC-DATA.
        03  MAS-DATA OCCURS 12 TIMES             PICTURE H9(10).
01  MAS-BUFFER.
        03 MAS-BUF     OCCURS 1796 TIMES        PICTURE H9(10).
PROCEDURE DIVISION.
MASGEN-OPEN-FILES.
    OPEN INPUT CARD-FILE.
    CALL 'OPENS' USING     MAS-UNIT          MAS-REC-SIZE
```

Figure A-8. Multi-Level Data Base Generation Program

```
                                   MAS-BLK-SIZE   MAS-BUFFER.
          READ-CARD-FILE.
              READ CARD-FILE AT END GO TO CLOSE-FILES.
              IF CARD-TYPE = 'A'  MOVE 101 TO MAS-REC-TYPE
                  PERFORM COUNTRY-ZERO-FILL
                  MOVE COUNTRY-CARD-DATA TO MAS-REC-DATA
                  GO TO WRITE-MAS-RECORD.
              IF CARD-TYPE = 'B'  MOVE 201 TO MAS-REC-TYPE
                  PERFORM STATE-ZERO-FILL THRU COUNTY-FILL
                  MOVE STATE-OR-COUNTY-DATA TO MAS-REC-DATA
                  GO TO WRITE-MAS-RECORD.
              IF CARD-TYPE = 'C'  MOVE 301 TO MAS-REC-TYPE
                  PERFORM COUNTY-FILL
                  MOVE STATE-OR-COUNTY-DATA TO MAS-REC-DATA
                  GO TO WRITE-MAS-RECORD.
          INVALID-INPUT-CARD.
              DISPLAY 'INVALID REC TYPE '  STATE-OR-COUNTY-CARD.
              GO TO READ-CARD-FILE.
          COUNTRY-ZERO-FILL.
              IF POPULATION-COUNTRY IS NOT EQUAL TO SPACES
               EXAMINE POPULATION-COUNTRY REPLACING LEADING SPACES BY ZERO.
              IF AREA-COUNTRY IS NOT EQUAL TO SPACES
               EXAMINE AREA-COUNTRY REPLACING LEADING SPACES BY ZERO.
          STATE-ZERO-FILL.
              IF STATE-RANK-IN-POP IS NOT EQUAL TO SPACES
               EXAMINE STATE-RANK-IN-POP REPLACING LEADING SPACES BY ZERO.
              IF STATE-RANK-IN-AREA IS NOT EQUAL TO SPACES
               EXAMINE STATE-RANK-IN-AREA REPLACING LEADING SPACES BY ZERO.
          COUNTY-FILL.
              IF POPULATION-STATE-COUNTY IS NOT EQUAL TO SPACES
               EXAMINE POPULATION-STATE-COUNTY
                         REPLACING LEADING SPACES BY ZERO.
              IF AREA-STATE-COUNTY IS NOT EQUAL TO SPACES
               EXAMINE AREA-STATE-COUNTY REPLACING LEADING SPACES BY ZERO.
          WRITE-MAS-RECORD.
              ADD 1 TO MAS-REC-NBR.
              CALL 'WRITES' USING MAS-UNIT MAS-REC-NBR MAS-RECORD.
              GO TO READ-CARD-FILE.
          CLOSE-FILES.
              CLOSE CARD-FILE.
              CALL 'CLOSEM' USING MAS-UNIT.
              STOP RUN.
```

Figure A-8. Multi-Level Data Base Generation Program (Continued)



NAME NATIØNS DICTIØNARY

ACCOUNT | PHONE

FILE NAME CARD

SHEET 1 OF 3

DATE

1 ACT	2 TYPE	3 FILE NAME	12 NBR DATA BASE RECORDS	20 MAX REC SIZE	24 RECS PER BLOCK	28 NBR LEVEL	30 SECURITY KEY	35 BLANK
I	F	NATIØNS	10000	13	137	3		

PASSWORD CARD

1 ACT	2 TYPE	3 PASSWORD	15 UPIND	16 BLANK
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			

RECORD IDENTIFIER CARD

1 ACT	2 TYPE	3 REC TYPE	6 REC ID	9 START LOC	13 END LOC	17 REC SIZE	21 BLANK
I	L	101	101	1	3	13	
I	L	201	201	1	3	13	
I	L	301	301	1	3	13	
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						

MSFC - Form 432-2 (Rev. May 1975)

Figure A-9. Multi-Level Data Base Dictionary Cards

NAME NATIØNS DICTIØNARY

ACCOUNT | PHONE

FIELD DEFINITION CARD 1

SHEET 2 OF 3

DATE

1	2	3 6	7	8	20	56	59	63	67	69	71	72	73	75	76	77	80
ACT	TYPE	FIELD NBR	NBR	FIELD NAME	REPORT TITLE	REC TYPE	START LOC	END LOC	NBR OCC	SRCH TYPE	INDEX	UP IND	DATA TYPE	DECS IN	DECS OUT	TLU TABLE NBR	GTLU
I	M	110	1	CØUNTRY	CØUNTRY	101	7	30		CH	Y	N	0				
I	M	120	1	PRES	PRESIDENT	101	31	54		CH	N	N	1				
I	M	130	1	N-PØP	PØPULATIØN ØF CØUNTRY	101	55	64			Y	Y	2				
I	M	140	1	N-AREA	AREA ØF CØUNTRY	101	65	72			N	Y	2				
I	M	150	1	CØN	CØNTINENT	101	73	73			N	N	2			1	
I	M	210	1	STATE	STATE	201	7	20		CH	Y	N	1				
I	M	220	1	CAP	CAPITAL	201	21	36		CH	N	N	0				
I	M	230	1	S-PØP	PØPULATIØN ØF STATE	201	37	44			Y	Y	2				
I	M	240	1	S-AREA	AREA ØF STATE	201	45	50			N	N	2				
I	M	250	1	PRANK	RANK IN PØP	201	51	52			N	Y	2				
I	M	260	1	ARANK	RANK IN AREA	201	53	54			N	N	2				
I	M	270	1	GØV	GØVERNØR	201	55	78			N	Y	1				
I	M	310	1	CØUNTY	CØUNTY	301	7	20		CH	Y	N	0				
I	M	320	1	SEAT	CØUNTY SEAT	301	21	36		CH	N	N	0				
I	M	330	1	C-PØP	PØPULATIØN ØF CØUNTY	301	37	44			N	Y	1				
I	M	340	1	C-AREA	AREA ØF CØUNTY	301	45	50			N	Y	2				
I	M	9101	1	ACARD	A CARD	101	1	3			N	N	2				
I	M	9201	1	BCARD	B CARD	201	1	3			N	N	2				
I	M	9301	1	CCARD	C CARD	301	1	3			N	N	2				
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														

MSFC - Form 4?2-4 (Rev. May 1975)

Figure A-9. Multi-Level Data Base Dictionary Cards (Continued)

NAME NATIØNS DICTIØNARY	
ACCOUNT	PHONE

TABLE LOOKUP CARD

SHEET 3 OF 3	
DATE	

1	2	3	6	15	63
ACT	TYPE	TLU TABLE NBR	TLU DATA BASE VALUE	TLU – REPORT – VALUE	BLANK
I	T	1	1	ASIA	
I	T	1	2	AFRICA	
I	T	1	3	NØRTH AMERICA	
I	T	1	4	SØUTH AMERICA	
I	T	1	5	ANTARCTICA	
I	T	1	6	EURØPE	
I	T	1	7	AUSTRALIA	
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				

Figure A-9. Multiple-Level Data Base Dictionary Cards (Continued)

```
  06 02 75             MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING         PAGE   1

  FILE NAME = NATIONS         RECORD SIZE (WDS) =    13    BLOCK SIZE =   137    LEVELS =   3    FILE SECURITY =

**WARNING**  NO PASSWORD CARDS HAVE BEEN ENTERED FOR THIS DICTIONARY


  06 02 75             MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING         PAGE   1

  FILE NAME = NATIONS         RECORD SIZE (WDS) =    13    BLOCK SIZE =   137    LEVELS =   3    FILE SECURITY =

       DICTIONARY RECORD TYPE         RECORD IDENTIFIER         START LOCATION      END LOCATION       RECORD SIZE (WDS)

                 101                         101                     0001              0003                   0013
                 201                         201                     0001              0003                   0013
                 301                         301                     0001              0003                   0013

   FIELD       FIELD DICT LOCATION  FIELD I  UP              REPORT FIELD TITLE                  TITLE  DEC DATA SRCH     TABLE LOOKUP
   NAME         NO   LVL  START END  SIZE N DATE                                                  CHARS  I O TYPE TYPE   NO   WIDTH  GLOBAL
                     CODE                 D                                                              N                     (MAX)

COUNTRY        0110  101  0007 0030  024  Y         COUNTRY                                        07    0 0  00   CH
PRES           0120  101  0031 0054  024            PRESIDENT                                      09    0 0  01   CH
N-POP          0130  101  0055 0064  010  Y   Y     POPULATION OF COUNTRY                          21    0 0  02   RG
N-AREA         0140  101  0065 0072  008      Y     AREA OF COUNTRY                                15    0 0  02   RG
CON            0150  101  0073 0073  001            CONTINENT                                      09    0 0  02   RG    001    13
STATE          0210  201  0007 0020  014  Y         STATE                                          05    0 0  01   CH
CAP            0220  201  0021 0036  016            CAPITAL                                        07    0 0  00   CH
S-POP          0230  201  0037 0044  008  Y   Y     POPULATION OF STATE                            19    0 0  02   RG
S-AREA         0240  201  0045 0050  006            AREA OF STATE                                  13    0 0  02   RG
PRANK          0250  201  0051 0052  002      Y     RANK IN POP                                    11    0 0  02   RG
ARANK          0260  201  0053 0054  002            RANK IN AREA                                   12    0 0  02   RG
GOV            0270  201  0055 0078  024      Y     GOVERNOR                                       08    0 0  01   RG
COUNTY         0310  301  0007 0020  014  Y         COUNTY                                         06    0 0  00   CH
SEAT           0320  301  0021 0036  016            COUNTY SEAT                                    11    0 0  00   CH
C-POP          0330  301  0037 0044  008      Y     POPULATION OF CITY                             18    0 0  02   RG
C-AREA         0340  301  0045 0050  006      Y     AREA OF CITY                                   12    0 0  02   RG
ACARD          9101  101  0001 0003  003            A CARD                                         06    0 0  02   RG
BCARD          9201  201  0001 0003  003            B CARD                                         06    0 0  02   RG
CCARD          9301  301  0001 0003  003            C CARD                                         06    0 0  02   RG
```

Figure A-10. Multi-Level Data Base Dictionary Listing

```
06 02 75              MARSHALL INFORMATION RETRIEVAL AND DISPLAY SYSTEM (MIRADS) DICTIONARY FILE LISTING         PAGE   2

FILE NAME = NATIONS          RECORD SIZE (WDS) =    13    BLOCK SIZE =   137    LEVELS =   3    FILE SECURITY =

     TABLE NO     DATABASE VALUE     REPORT TITLE

        001             1               ASIA
        001             2               AFRICA
        001             3               NORTH AMERICA
        001             4               SOUTH AMERICA
        001             5               ANTARCTICA
        001             6               EUROPE
        001             7               AUSTRALIA
```

Figure A-10. Multi-Level Data Base Dictionary Listing (Continued)

APPENDIX B - DICTIONARY CODING FORMS

NAME	
ACCOUNT	PHONE

FILE NAME CARD

SHEET ________	OF ________
DATE ________	

1	2	3	12	20	24	28	30	36
ACT	TYPE	FILE NAME	NBR DATA BASE RECORDS	MAX REC SIZE	RECS PER BLOCK	NBR LEVEL	SECURITY KEY	BLANK
	F							

PASSWORD CARD

1	2	3	15	16
ACT	TYPE	PASSWORD	UPIND	BLANK
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			
	I			

RECORD IDENTIFIER CARD

1	2	3	6	9	13	17	21
ACT	TYPE	REC TYPE	REC ID	START LOC	END LOC	REC SIZE	BLANK
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						
	L						

MSFC - Form 432-2 (Rev. May 1975)

NAME	
ACCOUNT	PHONE

FIELD DEFINITION CARD 1

SHEET	OF
DATE	

1	2	3 6	7	8	20	56	59	63	67	69	71	72	73	75	76	77	80
ACT	TYPE	FIELD NBR	NBR	FIELD NAME	REPORT TITLE	REC TYPE	START LOC	END LOC	NBR OCC	SRCH TYPE	INDEX	UP IND	DATA TYPE	DECS IN	DECS OUT	TLU TABLE NBR	GTLU
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														
	M		1														

MSFC - Form 432-4 (Rev. May 1975)

NAME	
ACCOUNT	PHONE

TABLE LOOKUP CARD

SHEET ______ OF ______	
DATE	

1	2	3	6	15	63
ACT	TYPE	TLU TABLE NBR	TLU DATA BASE VALUE	TLU — REPORT — VALUE	BLANK
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				
	T				

MSFC - Form 432 (Rev. May 1971)